Visual Rules Suite - Execution Platform

Execution Server Benutzerhandbuch

Version 6.0.0

Bosch Software Innovations



Americas:
Bosch Software Innovations Corp.
161 N. Clark Street
Suite 3500
Chicago, Illinois 60601/USA
Tel. +1 312 368-2500
info@bosch-si.com
www.bosch-si.com

Asia:
Bosch Software Innovations
c/o Robert Bosch (SEA) Pte Ltd
11 Bishan Street 21
Singapore 573943
Tel. +65 6571 2220
info-sg@bosch-si.com
www.bosch-si.com

Europe:
Bosch Software Innovations GmbH
Ziegelei 7
88090 Immenstaad/GERMANY
Tel. +49 7545 202-300
info-de@bosch-si.com
www.bosch-si.de

Execution Server Benutzerhandbuch: Version 6.0.0

Visual Rules Execution Server 6.0.0



Inhaltsverzeichnis

Kapitel 1. Einleitung

1.1. Anwendungsbereich

Der *Visual Rules Execution Server* ist eine Web-Anwendung, die es ermöglicht, mehrere Versionen von Regeln zu verwalten und diese als Web-Services auszuführen. Darüberhinaus erlaubt er das Austauschen und Deployen von Regeln zur Laufzeit und sammelt Statistiken zu den ausgeführten Regeln.

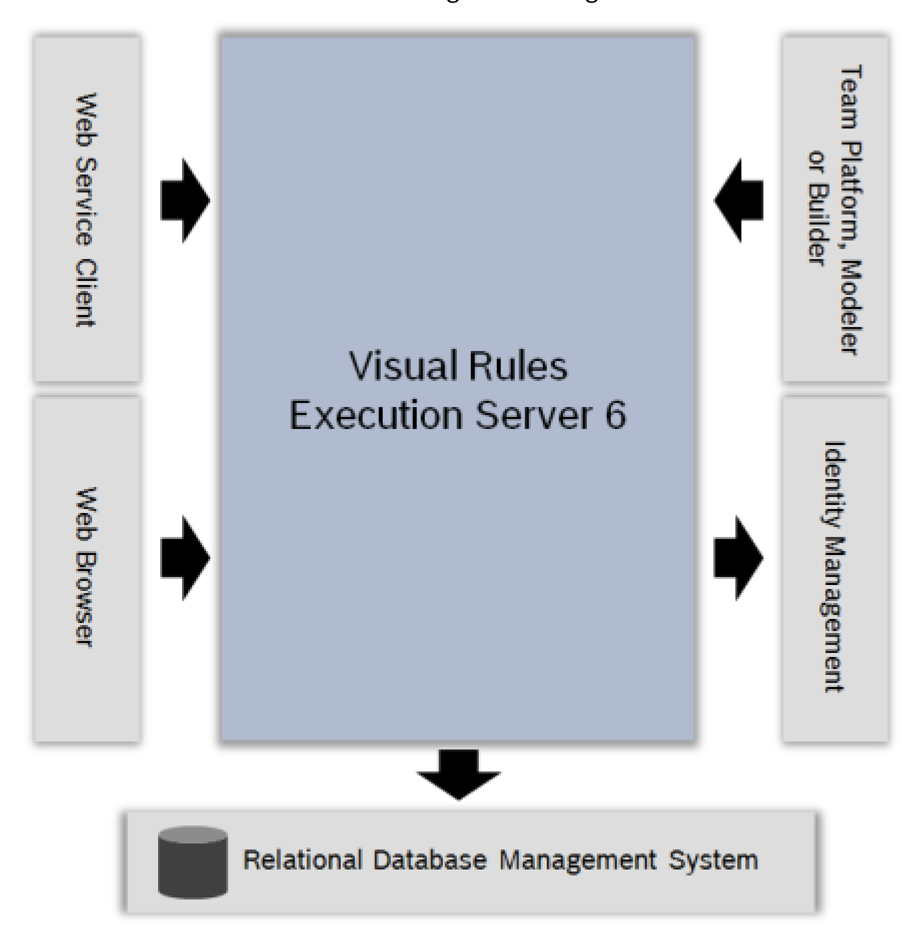


Abbildung 1.1. Überblick Execution Server 6

Der erste Schritt um Regeln als Web-Service auszuführen ist, diese zu bauen und als Regelbibliothek zu paketieren. Danach können diese Bibliotheken auf dem *Visual Rules Execution Server* bereitgestellt werden. Dies kann entweder mit dem *Visual Rules Modeler*, der *Team Platform* oder dem *Builder* erreicht werden, die alle eine entsprechende Integration bereitstellen.

Regelbibliotheken werden mit der Webkonsole verwaltet. Diese ist eine Web-Oberfläche, auf die Benutzer mit Standard-Webbrowsern zugreifen können. Wurde eine Regelbibliothek und ihre Abhängigkeiten bereitgestellt, so wird daraus ein sogenannter Rule Service erzeugt, welcher als Web-Service mittels Standard Web Service Clients aufgerufen werden kann. Jede Ausführung eines Rule Service kann eine Statistik erzeugen, welche im *Visual Rules Modeler* angezeigt werden kann. Zur Persistierung von Statistiken und Bibliotheken wird ein relationales Datenbankverwaltungssystem (RDBMS) verwendet.

Während all dieser Schritte ist eine Authentifizierung notwendig und Zugriffe sind durch Berechtigungen eingeschränkt, welche vom Identity Management verwaltet werden.

Bei alledem unterstützt der *Execution Server* Mandantenfähigkeit.

1.2. Identity Management

Das Identity Management (IM) ist eine Benutzerverwaltungskomponente. Ihre Bedienoberflächen können von anderen Systemen verwendet werden, um ihre Benutzer zu identifizieren und zu authorisieren. Der wesentliche Anwendungsbereich des IM in einer Kundenanwendung besteht in der Verwaltung von Berechtigungen.



Sie können das Identity Management in der Webkonsole über den Tab **Benutzerverwaltung** erreichen, wenn nötig. Weitere Informationen zum Identity Management finden Sie in der zugehörigen Dokumentation.

1.3. Berechtigungs-Konzept

Das Berechtigungs-Konzept des Visual Rules Execution Server basiert auf den folgenden Konzepten:

- Abschnitt 1.3.1, „Execution-Server-Berechtigungen“
- Abschnitt 1.3.2, „Execution-Server-Benutzerrollen“

Die tatsächlichen Rechte eines Benutzers/Teams ergeben sich aus deren Kombination.



Ein Team kann weitere Teams enthalten. Generell sind alle Berechtigungen, die einem Team zugewiesen sind, automatisch auch jedem Benutzer/Team innerhalb des Teams zugewiesen.

1.3.1. Execution-Server-Berechtigungen

Execution-Server-Berechtigungen sind Berechtigungen, die im Identity Management mit Bezug auf den Visual Rules Execution Server definiert sind. Dabei repräsentiert jede Berechtigung bestimmte Arten von Aktionen.



Standardmäßig wird jedem Mandanten das Standard-Berechtigungspaket zugewiesen.

Die folgenden Execution-Server-Berechtigungen werden unterschieden:

Tabelle 1.1. Übersicht der Execution-Server-Berechtigungen

Berechtigung	Beschreibung
Visual Rules Archiv bereitstellen	Berechtigung zum Bereitstellen eines Rule Service und aller erforderlichen Bibliotheken mittels Visual Rules Archiv
(Regel-)Bibliothek bereitstellen	Berechtigung zum Bereitstellen eines Rule Service mittels Regelbibliothek oder einer Bibliothek
(Regel-)Bibliothek löschen	Berechtigung zum Löschen eines bereitgestellten Rule Service oder einer Bibliothek
Log-Informationen anzeigen/herunterladen	Berechtigung zum Anzeigen/Herunterladen von Log-Informationen
Logging konfigurieren	Berechtigung zum Ändern des Detaillierungsgrads des Logging
Lizenzen hinzufügen/löschen	Berechtigung zum Hinzufügen/Löschen von Lizenzen
Applikationskonfiguration anzeigen	Berechtigung zum Anzeigen der Applikationskonfiguration

1.3.1.1. Standard-Berechtigungspaket

Das Standard-Berechtigungspaket wird jedem Mandanten standardmäßig zugewiesen. Es enthält die folgenden Execution Server-Berechtigungen:

- Visual Rules Archiv bereitstellen
- (Regel-)Bibliothek bereitstellen

- (Regel-)Bibliothek löschen

1.3.2. Execution-Server-Benutzerrollen

Execution-Server-Benutzerrollen sind Benutzerrollen, die im Identity Management mit Bezug auf den Visual Rules Execution Server definiert sind. Dabei repräsentiert jede Benutzerrolle einen virtuellen Benutzer, der befugt ist, bestimmte Aufgaben auszuführen (d.h. dem die entsprechenden Execution-Server-Berechtigungen standardmäßig per Definition zugewiesen sind).

Standardmäßig werden die folgenden Execution-Server-Benutzerrollen bereitgestellt:

- Abschnitt 1.3.2.1, „Administrator“
- Abschnitt 1.3.2.2, „Deployer“

Weitere Execution-Server-Benutzerrollen können im Identity Management definiert werden.



Standardmäßig beinhaltet jede Execution-Server-Benutzerrolle die Berechtigungen *Query users* und *Query groups*, die im Identity Management allgemein (d.h. mit Bezug auf alle Execution Server Anwendungen) definiert sind. Diese Berechtigungen sind die Mindestvoraussetzung, um über den Execution Server mit anderen Visual Rules Produkten zu arbeiten.

1.3.2.1. Administrator

Die Benutzerrolle *Administrator* repräsentiert einen Administrator. Ihm sind standardmäßig alle Execution-Server-Berechtigungen zugewiesen.

1.3.2.2. Deployer

Die Benutzerrolle *Deployer* repräsentiert einen Benutzer mit allen relevanten Execution-Server-Berechtigungen zum Bereitstellen von Rule Services und Bibliotheken. Dies sind die Folgenden:

- Visual Rules Archiv bereitstellen
- (Regel-)Bibliothek bereitstellen
- (Regel-)Bibliothek löschen

1.4. Mandantenfähigkeit und Konzept der Datentrennung

1.4.1. Mandantenfähigkeit

Die Mandantenfähigkeit erlaubt die Nutzung einer einzelnen Execution Server Instanz und zugehöriger Datenbank durch mehrere Mandanten. Der Execution Server stellt sicher, dass die Daten eines jeden Mandanten nur durch die Benutzer des besitzenden Mandanten zugreifbar sind. Im Gegensatz zu dem Ansatz, jedem Mandanten eine eigene Instanz zur Verfügung zu stellen, ermöglicht Mandantenfähigkeit einen effizienteren Umgang mit Ressourcen wie z.B. Hardware- oder Virtualisierungsinfrastrukturen und Lizenzen.

1.4.2. Konzept der Datentrennung

Das Konzept der Datentrennung des Visual Rules Execution Server beinhaltet folgende Aspekte:

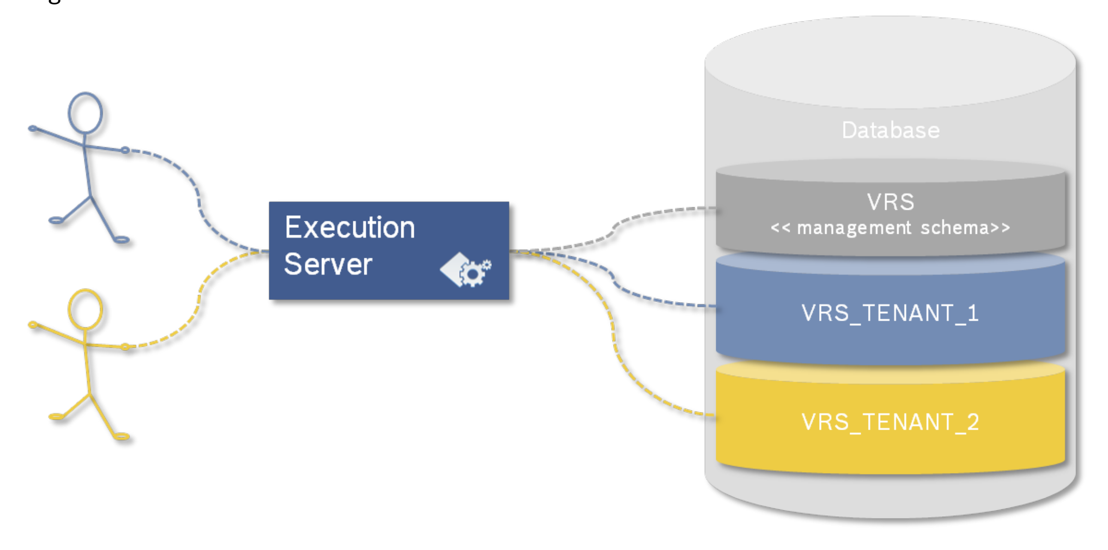
- Abschnitt 1.4.2.1, „Trennung der Mandantendaten“
- Abschnitt 1.4.2.2, „Trennung der Anwendungsdaten“



Zu den Gründen für die Datentrennung siehe Abschnitt 1.4.2.3, „Notwendigkeit und Vorteile der Datentrennung“.

1.4.2.1. Trennung der Mandantendaten

Da der Execution Server Mandantenfähigkeit unterstützt, muss sichergestellt sein, dass die Daten der Mandanten (z.B. Bibliotheken, Ausführungen, usw.) getrennt abgelegt werden. Dazu braucht jeder Mandant einen eigenen "privaten" Bereich für seine Daten. Dies wird erreicht, indem für jeden Mandanten ein eigenes Datenbankschema verwendet wird. Darüberhinaus wird ein zusätzliches Datenbankschema benötigt, um globale (d.h. nicht mandantenspezifische) Information zu speichern, so wie das Mapping zwischen den Mandanten und den ihnen zugewiesenen Datenbankschemata.



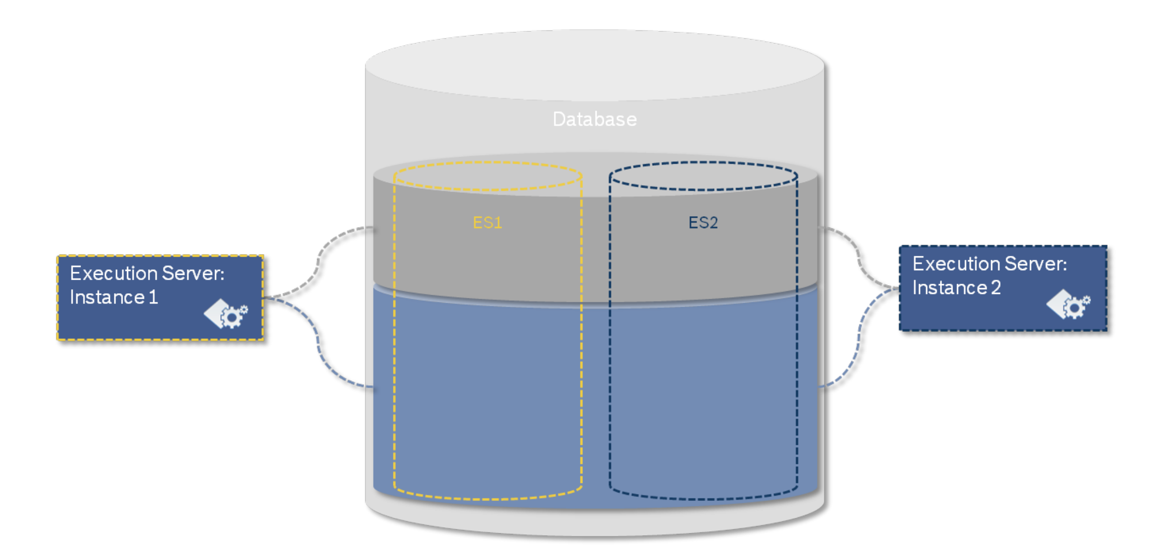
1.4.2.2. Trennung der Anwendungsdaten

Im Gegensatz zum herkömmlichen Ansatz, für jede Anwendung ein eigenes Datenbankschema zu verwenden, verwendet der Visual Rules Execution Server für jeden Mandanten ein eigenes Datenbankschema. Der Grund hierfür liegt in der Tatsache, dass die Trennung der Mandantendaten erste Priorität hat. Anwendungen trennen ihre Daten durch die Verwendung einer Art von Namensraum. Ein eigener Namensraum für jede Anwendung ermöglicht die gemeinsame Verwendung eines Mandantenschemas durch verschiedene Anwendungen. So wird sichergestellt, dass eine Anwendung nicht die Daten anderer Anwendungen beeinflußt (nicht einmal, wenn diese vom gleichen Typ sind, wie z.B. zwei verschiedene Instanzen des Execution Servers). Die sogenannten Namensräume sind lediglich Präfixe, welche den Namen der Datenbankobjekte einer Anwendung vorangestellt sind. Das folgende Beispiel zeigt dies für die Verwendung der Tabelle EP_VERSION.

Tabelle 1.2. Trennung der Anwendungsdaten

Applikationsname	Namensraum (Präfix)	DB-Objektname	Präfix+'_'+DB-Objektname
Execution Server 1	ES1	EP_VERSION	ES1_EP_VERSION
Execution Server 2	ES2	EP_VERSION	ES2_EP_VERSION

Dieses Muster wird über alle betroffenen Schemata (globale Verwaltungsschemata und Mandantenschemata) hinweg auf alle Datenbankobjekte angewandt, die spezifisch sind für eine Anwendungsinstanz. Die folgende Abbildung zeigt, wie die Namensräume eine Art virtueller anwendungsspezifischer Schemata über mehrere tatsächliche Datenbankschemata aufspannen:



1.4.2.3. Notwendigkeit und Vorteile der Datentrennung

Herkömmliche Mandantenfähigkeit setzt voraus, dass die Anwendung die Daten der Mandanten trennt und sicherstellt, dass kein Mandant auf die Daten anderer Mandanten zugreifen kann. Dies kann auf verschiedene Arten geschehen, z.B. durch eine zusätzliche Spalte in jeder Tabelle, welche die ID des Mandanten enthält. Der Ansatz des Visual Rules Execution Server hat folgende Vorteile:

- Mandantendaten können physikalisch getrennt gespeichert werden, indem verschiedene Tablespaces verwendet werden.
- Ressourcen können effizient genutzt werden, da mehrere Anwendungen der gleichen Art in einem einzelnen Mandantenschema installiert sein können.
- Eine Schema-Migration ist für eine bestimmte Anwendung eines bestimmten Mandanten möglich (z.B. während einer Aktualisierung der Anwendung).
- Die Datenzusammensetzung eines einzelnen Mandanten betrifft nicht notwendigerweise andere Mandanten.

1.5. Distribution

Der *Visual Rules Execution Server* ist als Web Archive (WAR) gepackt und wird als Bestandteil einer Distribution ausgeliefert. Die Distribution ist ein Zip Archiv und enthält folgendes:

Tabelle 1.3. Distributions Inhalt

Ordner	Inhalt
api-source	Quellcode des *Application Programming Interface (API)*. Hilfreich sofern ein benutzerdefinierter Metadaten Mapper für den Execution Server entwickelt wird.
legalnotice	Rechtliche Informationen zu Lizenzen und Verwendung von Bibliotheken von Drittanbietern
maintenance	Maintenance Tool
doc	Handbücher

1.5.1. Maintenance Tool

Das Maintenance Tool ist ein Kommandozeilenprogramm für den *Visual Rules Execution Server* um Wartungsaufgaben auszuführen. Es kann verwendet werden, um Tabellen in einem leeren Datenbank Schema anzulegen. Es ist Bestandteil der Execution Server Distribution und da es als Zip Archiv paketiert ist, muss es als

erstes entpackt werden, damit es verwendet werden kann. Zum Aufruf des Programms dient entweder ein `*.cmd` (für Microsoft Windows Betriebssysteme) oder ein `*.sh` (für *nix Betriebssystem) Skript. Die Verbindung zur Datenbank verwendet JDBC und benötigt aus diesem Grund den JDBC Treiber für die Datenbank. Die jar-Datei des Treibers muss dazu nur in das `driver` Verzeichnis kopiert werden, dass sich im Verzeichnis des entpackten Archivs befindet.

Nach dem Entpacken kann das Programm auf der Kommandozeile ausgeführt werden. Dazu öffnen Sie eine Kommandozeile und navigieren zu dem Verzeichnis, in dem das Zip Archiv entpackt wurde. Im folgenden Beispiel wird angenommen, dass es sich um das Verzeichnis `maintenance` in `tmp` auf einem Microsoft Windows Betriebssystem handelt:

```
C:\tmp\maintenance>
```

Das Verzeichnis ist richtig, wenn sich das Kommandozeilenprogramm ausführen lässt, beispielsweise durch Eingabe von:

```
MaintenanceTool
```

Dieser Aufruf wird alle verfügbaren Kommandozeilen-Optionen mit ihrer Beschreibung und Alternativen ausgeben.

Kapitel 2. Aufsetzen des Execution Servers

Voraussetzungen: Die erforderliche Datenbank sowie das Identity Management sind gestartet.

Das Aufsetzen des Execution Servers umfasst folgende Tätigkeiten:

- Abschnitt 2.1, „Vorbereiten der Datenbankumgebung“
- Abschnitt 2.2, „Vorbereiten des Web Servers“
- Abschnitt 2.3, „Deployment des Execution Servers“
- Abschnitt 2.4, „Installation des Execution Servers“
- Abschnitt 2.6, „Installation der Lizenz“
- Abschnitt 2.7, „Konfiguration des Execution Servers“



Es ist nicht möglich eine ältere Version des Execution Server auf die aktuelle Version zu migrieren.

2.1. Vorbereiten der Datenbankumgebung

Der Visual Rules Execution Server unterstützt bezüglich der Datenbank im Wesentlichen zwei Betriebsmodi, nämlich einen automatischen Modus und einen manuellen Modus. Welcher Betriebsmodus passend ist, hängt hauptsächlich von Ihren Sicherheitsanforderungen ab.



Im automatischen Modus benötigt der Datenbankbenutzer Berechtigungen, um die Struktur der Datenbank zu ändern. Der automatische Modus ist nicht für produktive Umgebungen geeignet, ist aber dennoch wertvoll für Evaluationszwecke, da er den Systembetrieb stark vereinfacht. Bei Verwendung des automatischen Modus wird empfohlen, für die Installation des Execution Server eine separate Datenbankinstanz bereit zu stellen. Bei Verwendung des manuellen Modus müssen die Datenbankschemata für Mandanten manuell aufgesetzt werden. Für Beispiele sehen Sie Abschnitt 2.1.1.1, „Beispiel: Oracle Datenbank“ und Abschnitt 2.1.2.1, „Beispiel: Oracle Datenbank“.

2.1.1. Automatischer Modus



Bei Verwendung des automatischen Modus wird empfohlen, für die Installation des Execution Server eine separate Datenbankinstanz bereit zu stellen.

Im automatischen Modus wird ein Großteil der administrativen Aufgaben vom Execution Server durchgeführt (z.B. das Erzeugen der Mandantenschemata und Ihre Initialisierung mit Datenbankobjekten wie Tabellen). Um dies tun zu können, muss der Datenbankbenutzer, der vom Execution Server für die Datenbankverbindung verwendet wird, zu folgenden Aktionen berechtigt sein:

- Erzeugen neuer Schemata in der Datenbank
- Initialisieren dieser Schemata mit Tabellen und anderen Datenbankobjekten
- Ausführen regulärer Datenbankoperationen auf diesen Objekten

2.1.1.1. Beispiel: Oracle Datenbank

Wenn der Execution Server im automatischen Modus betrieben werden soll, muss ein Datenbankadministrator die folgenden Statements ausführen, um die Datenbank für die Installation des Execution Server vorzubereiten:

```
-- Create a database user which the Execution Server uses to connect to the
-- database. This user must have sufficient privileges for regular
-- operation of the Execution Server and additionally to create database schemata
-- for new tenants automatically (the specification of the tablespaces
-- in the example is optional).

create user <username>
identified by <password>
default tablespace <tablespaceName>
temporary tablespace TEMP;

-- grant login privilege so the Execution Server is able to connect to the
-- database

grant create session to <username>;

-- grant create user privilege so that the Execution Server is capable to create
-- a database schema for the tenant (the schema is created when a user logs
-- into the Execution Server for a tenant for the first time if no schema
-- exists for the tenant, i.e. a schema named according to the tenants
-- technical name prefixed with 'VRS_')

grant create user to <username>;

-- grant create database object privileges in all schemata in the database
-- so that the Execution Server is able to automatically create its database
-- objects in the tenant schemata

grant create any table to <username>;
grant create any view to <username>;
grant create any index to <username>;
grant create any sequence to <username>;
grant create any trigger to <username>;

-- grant CRUD privileges in all schemata so that the Execution Server is able to
-- perform regular database operations across all tenant schemata

grant select any table to <username>;
grant insert any table to <username>;
grant update any table to <username>;
grant delete any table to <username>;
```

Beispiel 2.1. Automatischer Modus in Oracle Datenbank

2.1.2. Manueller Modus



Bei Verwendung des manuellen Modus müssen die Datenbankschemata für Mandanten manuell aufgesetzt werden. Siehe Abschnitt 2.5, „Manuelles Aufsetzen der Mandanten“.

Der manuelle Betriebsmodus setzt voraus, dass ein Datenbankadministrator den Benutzer, der vom Execution Server für die Datenbankverbindung verwendet wird, anlegt und konfiguriert.

2.1.2.1. Beispiel: Oracle Datenbank

```
-- Create a database user which the Execution Server uses to connect to the
-- database. This user only must have sufficient privileges for regular
-- operation of the Execution Server. In this operational mode the database
-- administrator is responsible for the creation of database schemata
-- and their initialization with the Execution Server database tables using
-- a command line based maintenance tool which is provided as part of
-- the Execution Server distribution.

create user <username>
identified by <password>
default tablespace <tableSpaceName>
temporary tablespace <tempTableSpaceName>
quota unlimited on <tableSpaceName>;

-- grant login privilege so the Execution Server is able to connect to the
-- database

grant create session to <username>;
```

Beispiel 2.2. Beispiel Oracle Datenbank: Anlegen und Konfigurieren eines Datenbankbenutzers für den manuellen Modus

2.2. Vorbereiten des Web Servers

Um den Web Server vorzubereiten, fügen Sie die Datenbank als Daten Resource hinzu, die über einen JNDI Namen erreichbar ist. Der JNDI Name ist abhängig vom Application Server Hersteller, wie im Folgenden beschrieben.

2.2.1. Vorbereiten eines Tomcat Application Servers

Um einen Tomcat Application Server vorzubereiten, konfigurieren Sie die Datenbank als eine JDBC Daten Resource mit dem JNDI Namen `java:jboss/jdbc/executionserverDS`.

2.2.2. Vorbereiten eines WebSphere Application Servers

Um einen WebSphere Application Server vorzubereiten, führen Sie folgende Schritte durch:

1. Konfigurieren Sie die Anmeldedaten für die Datenbank als **JAAS J2C Authentication Data Alias**.
2. Konfigurieren sie den für die Datenbank notwendigen herstellerspezifischen JDBC Provider.
3. Konfigurieren Sie die Datenbank als JDBC Datenquelle. Benutzen Sie hierbei die folgenden Werte:
   - Der JNDI Name ist `jdbc/executionserverDS`.
   - In den Sicherheitseinstllungen, spezifizieren Sie den Authentication Alias für container-managed- sowie und component-managed Authentication Alias.
   - In den Sicherheitseinstllungen, setzen Sie die Mapping-Konfiguration für den Alias auf `DefaultPrincipalMapping`.
4. Speichern Sie die Konfigurationsänderungen und starten Sie den WebSphere Server neu um den Authentication Alias effektiv werden zu lassen.

2.2.3. Vorbereiten eines JBoss Application Servers

Um einen JBoss Application Server vorzubereiten, konfigurieren Sie die Datenbank als eine JDBC Daten Resource mit dem JNDI Namen `java:jboss/jdbc/executionserverDS`.

2.3. Deployment des Execution Servers

Der Execution Server wird als gepacktes Web Archiv (WAR) ausgeliefert und kann installiert werden, indem er auf einem Application Server bereitgestellt wird. Die grundsätzliche Vorgehensweise um den Execution Server zu Deployen ist wie folgt:

1. Deployen Sie die executionserver.war

2. Starten Sie den Execution Server.



Nach der Bereitstellung auf dem Application Server sollte probehalber die Webkonsole aufgerufen werden. Dies ist in Abschnitt 4.1, „Aufruf der Webkonsole“ beschrieben.

Der Deploymentprozess ist abhängig vom Application Server Hersteller. Wenn Sie einen Tomcat Application Server einsetzen sind ist keine besondere Prozedur notwendig. Die Prozedur für einen WebSphere Application Server finden Sie unter Abschnitt 2.3.1, „Spezifische Prozedur für Deployment auf WebSphere“. Wenn Sie einen JBoss Application Server einsetzen, lesen Sie Abschnitt 2.3.2, „Spezifische Prozedur für Deployment auf JBoss“.

2.3.1. Spezifische Prozedur für Deployment auf WebSphere

Wenn Sie einen WebSphere Anwendungsserver benutzen, ist die Prozedur den Execution Server zu Deployen wie folgt:



Der Standard Klassenlader von *WebSphere* ist nicht geeignet für den Execution Server.

WebSphere kann den vorkonfigurierten JNDI Namen für die Datenbank des Execution Servers nicht auflösen.

1. Fügen Sie die executionserver.war WAR Datei als neue Enterprise Application hinzu. Mappen Sie den Context-Root, z.B. auf `/executionserver`.

2. Passen sie den Modul Klassenlader des Execution Server wie folgt an:

   - Gehen Sie zu **Anwendungen > Anwendungstypen > WebSphere-Unternehmensanwendungen** und klicken Sie auf die executionserver_war.

   - In der Sektion**Module**, klicken Sie auf **Module Verwalten**.

   - Klicken Sie auf das Modul **Visual Rules Execution Server SOA**.

   - Im Drop-down Menü **Reihenfolge der Klassenlader**, wählen Sie `Mit dem lokalen Klassenlader geladene Klassen zuerst (übergeordneter zuletzt).`

   - Speichern Sie die Konfigurationsänderung.

3. Um den JNDI Namen für den Execution Server in der Execution Server Propertiesdatei anzupassen, gehen Sie wie folgt vor:

   - Starten Sie die Execution Server Anwendung. Die Execution Server Konfigurationsdatei wird mit initialen Werten angelegt.

   - Stoppen sie die Execution Server Anwendung.

   - Die Execution Server Konfigurationsdatei befindet sich im Execution Server Home Verzeichnis (siehe Abschnitt 2.7.1, „Execution Server Home“) unter `config/executionserver.properties`. Passen Sie in dieser Datei folgendes Property an: `executionserver.jndi.name=jdbc/executionserverDS`

4. Starten Sie den Execution Server.

2.3.2. Spezifische Prozedur für Deployment auf JBoss

Wenn Sie einen JBoss Anwendungsserver benutzen, ist die Prozedur den Execution Server zu Deployen wie folgt:



JBoss kann den vorkonfigurierten JNDI Namen für die Datenbank des Execution Servers nicht auflösen.

1. Fügen Sie die executionserver.war WAR Datei als neue Enterprise Application hinzu. Mappen Sie den Context-Root, z.B. auf `/executionserver`.

2. Um den JNDI Namen für den Execution Server in der Execution Server Propertiesdatei anzupassen, gehen Sie wie folgt vor:

   - Starten Sie die Execution Server Anwendung. Die Execution Server Konfigurationsdatei wird mit initialen Werten angelegt.

   - Stoppen sie die Execution Server Anwendung.

   - Die Execution Server Konfigurationsdatei befindet sich im Execution Server Home Verzeichnis (siehe Abschnitt 2.7.1, „Execution Server Home“) unter `config/executionserver.properties`. Passen Sie in dieser Datei folgendes Property an: `executionserver.jndi.name=java:jboss/jdbc/executionserverDS`

3. Starten Sie den Execution Server.

2.4. Installation des Execution Servers

Wenn die benötigte Datenbank, das Identity Management und die Execution Server Anwendung gestartet sind, können Sie den Execution Server installieren. Die Installation des Execution Servers beinhaltet folgende Schritte:

- Die Datenbank aufsetzen.

- Den Execution Server im Identity Management registrieren.

Beide Schritte werden durch den Installations-Wizard unterstützt. Dies ist die empfohlene Vorgehensweise, welche beschrieben wird in Abschnitt 2.4.1, „Installieren des Execution Server mit dem Installationsassistenten“.

Wenn Sie die Datenbankoperationen überprüfen und manuell ausführen möchten, benutzen Sie das Maintenance Tool um die Datenbank anzulegen. Dies ist beschrieben in Abschnitt 2.4.2, „Installieren des Execution Servers mit dem Maintenance Tool“. Das Maintenance Tool unterstützt nicht das Registrieren des Execution Servers im Identity Management. Benutzen Sie für diesen Schritt in jedem Fall den Installations-Wizard wie beschrieben in Abschnitt 2.4.1, „Installieren des Execution Server mit dem Installationsassistenten“.

2.4.1. Installieren des Execution Server mit dem Installationsassistenten

Sie können den Visual Rules Execution Server mithilfe des Installationsassistenten installieren.



Beim Installieren des Execution Server mit dem Installationsassistenten werden die wesentlichen Installationsschritte automatisch durchgeführt.

Sobald der Execution Server bereitgestellt wurde, ist der Installationsassistent im Web Browser erreichbar mit einer Adresse im Format `<protocol>://<server>:<port>/<context-name>`, z.B. `http://localhost: 8080/executionserver-6.x.y`.

1. Öffnen Sie den Installationsassistenten.

2. Die Seite *Identity Management Verbindung* öffnet sich:



3. Geben Sie die URLs vom Backend und der Webkonsole des Identity Management in den entsprechenden Feldern an.

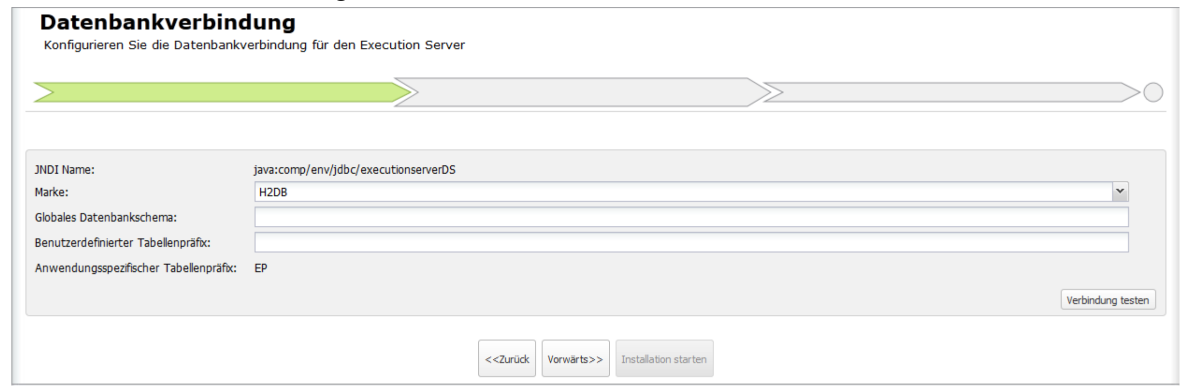
4. Geben Sie in den entsprechenden Feldern Benutzernamen, Passwort und Mandant für einen Administrator an, der berechtigt ist, Anwendungen im Identity Management zu installieren.



Die Execution Server Anwendung wird mit dem betreffenden Benutzer im Identity Management registriert.

5. Klicken Sie auf **Vorwärts>>**.

6. Die Seite *Datenbankverbindung* öffnet sich:



7. Füllen Sie die folgenden Felder oder ändern Sie ggfs. die Einträge:

   - **Marke**: Hersteller der Datenbank
   - **Globales Datenbankschema**: Name des Datenbankschemas, welches die Datenbanktabellen für die Schemaverwaltungskomponente, die von allen Visual Rules Suite Anwendungen verwendet wird.
   - **Benutzerdefinierter Tabellenpräfix**: Präfix der Execution Server-spezifischen Datenbanktabellen

8. Klicken Sie auf **Vorwärts>>**.

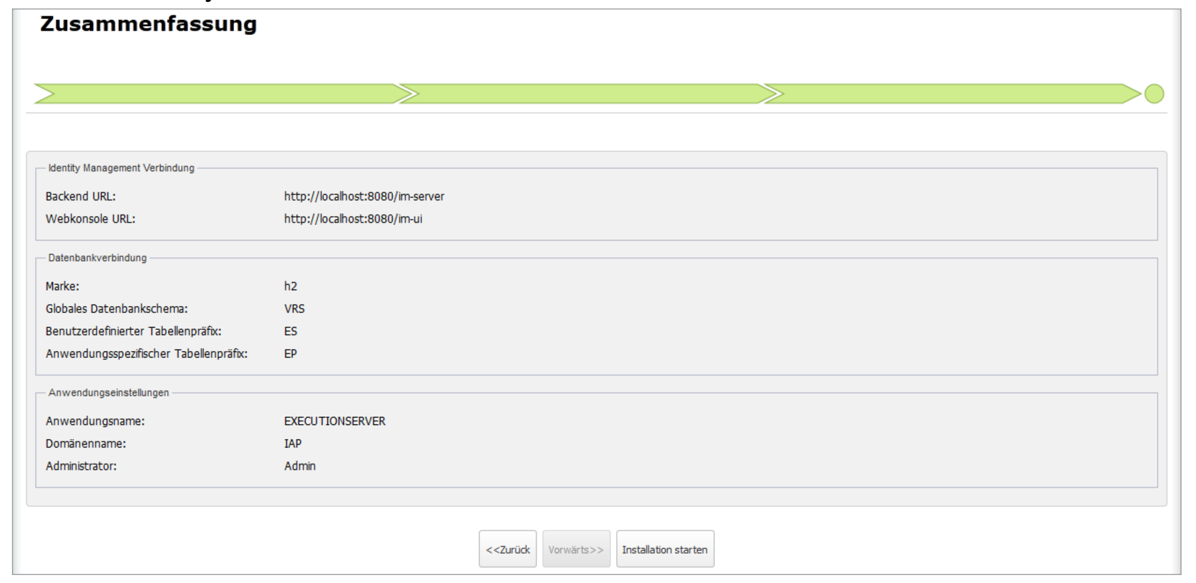
9. Die Seite *Anwendungseinstellungen* öffnet sich:



10. Füllen Sie die folgenden Felder oder ändern Sie ggfs. die Einträge:

   - **Domänenname**: Name der Domäne, unter der Ihr Execution Server im Identity Management installiert werden wird
   - **Anwendungsname**: Anwendungsname, mit dem Ihr Execution Server im Identity Management installiert werden wird
   - **Administrator**: Administrator, der berechtigt ist, Anwendungen im Identity Management zu installieren

11. Klicken Sie auf **Vorwärts>>**.
12. Die Seite *Summary* öffnet sich:



13. Klicken Sie auf **Installation starten**.
14. Der Execution Server wird installiert.

2.4.2. Installieren des Execution Servers mit dem Maintenance Tool

2.4.2.1. Anlegen der Datenbanktabellen für das globale Schema

Um die Datenbanktabellen für das globale Schema anzulegen, tun Sie Folgendes:

1. Extrahieren Sie die Distributionsdatei **Execution-Server-6.x.x-distribution.zip**.
2. Gehen Sie in der Kommandozeile zum Ordner **maintenance**.
3. Extrahieren Sie das Maintenance Tool Archiv **maintenanceTool.zip**.
4. Gehen Sie in der Kommandozeile in den extrahierten Maintenance Tool Ordner.

5. Starten Sie das Maintenance Tool, indem Sie eins der folgenden Kommandos verwenden (angepasst an Umgebung und Ziel der Installation):

   - **Windows:** MaintenanceTool.cmd --dumpGlobalSchemaDDL --executeSql --sqlDumpFile <dumpFileName> --globalSchemaName <globalSchemaName> --dbBrand <dbBrand> -dbDriverClass <driverClassName> -jdbcUrl <jdbcUrl> -dbUser <adminUserName> -dbPassword <adminUserPassword>

   - **Linux/Solaris :** MaintenanceTool.sh --dumpGlobalSchemaDDL --executeSql --sqlDumpFile <dumpFileName> --globalSchemaName <globalSchemaName> --dbBrand <dbBrand> -dbDriverClass <driverClassName> -jdbcUrl <jdbcUrl> -dbUser <adminUserName> -dbPassword <adminUserPassword>



Das Argument ‘executeSql’ ist optional und muss nur spezifiziert werden, wenn die Datenbanktabellen sofort angelegt werden sollen. Ohne dieses Argument würden die resultierenden SQL Statements lediglich in die durch das Argument ‘sqlDumpFile’ spezifizierte Datei ausgegeben und können dann später manuell ausgeführt werden (in diesem Fall müssen die zu JDBC gehörigen Argumente auch nicht definiert werden). Wenn das Argument spezifiziert wird, muss der für die jeweilige Datenbank geeignete Treiber in das Verzeichnis **driver** des Maintenance Tools gelegt werden.

Wenn das Maintenance Tool ohne irgendwelche Argumente gestartet wird, werden die möglichen Argumente auf die Konsole geschrieben. Falls manche der Argumente, die für das Ausführen einer bestimmten Aufgabe benötigt werden, nicht in der Kommandozeile spezifiziert sind, werden die benötigten Argumente über den Argumentbeschreibungen angezeigt.

Unter Verwendung des folgenden Kommandos können Sie eine Properties-Datei schreiben, wobei Sie Standardwerte für Argumente definieren können:

- **Windows:** MaintenanceTool.cmd --writeOptionProperties maintenance.properties

- **Linux/Solaris:** MaintenanceTool.sh --writeOptionProperties maintenance.properties

Sie können die Standardwerte wiederverwenden, indem Sie das Argument ‘readOptionProperties’ spezifizieren, wie im folgenden Beispiel gezeigt:

- **Windows:** MaintenanceTool.cmd --dumpExecutionServerDDL --dbBrand oracle -readOptionProperties maintenance.properties

- **Linux/Solaris:** MaintenanceTool.sh --dumpExecutionServerDDL --dbBrand oracle -readOptionProperties maintenance.properties

Im obigen Beispiel würden die Werte für die Argumente ‘tenantSchemaName’ und ‘tablePrefix’ aus der Datei übernommen (vorausgesetzt, dass sie dort definiert sind). Der Wert für das Argument ‘dbBrand’ würde, unabhhängig davon, ob das Argument in der Datei spezifiziert wurde, aus der Kommandozeile übernommen, da die in der Kommandozeile spezifizierten Argumente Vorrang vor den Werten aus der Datei haben.

6. Weisen Sie dem Datenbankbenutzer, der vom Execution Server für die Datenbankverbindung verwendet wird, Berechtigungen für die Datenbankobjekte im globalen Schema zu, indem Sie die Statements ausführen, die zum Beispiel für eine Oracle Datenbank durch folgende Anfrage erzeugt werden:

```
-- Take the result of this select statement and execute it, to grant the necessary privileges
select grant_statements||owner||'.'||db_obj.object_name||' to <username>;' as grant_statements
from (
select 'grant select on ' as grant_statements, v.owner, v.VIEW_NAME as object_name
from sys.all_views v
union all
select 'grant select,insert,update,delete on ',
t.owner, t.table_name
from sys.all_tables t
) db_obj
where db_obj.owner = '<globalSchemaName>'
```

Beispiel 2.3. Beispiel-Anfrage Oracle Datenbank

2.5. Manuelles Aufsetzen der Mandanten



Ein manuelles Aufsetzen der Mandanten ist nur nötig, wenn bezüglich der zugrundeliegenden Datenbank der manuelle Modus verwendet wird. Im automatischen Modus passiert dies automatisch.

Um manuell neue Mandanten anzulegen, tun Sie Folgendes:

1. Erzeugen Sie mithilfe des Execution Server Maintenance Tools ein neues Mandantenschema, das aus Schema-Präfix 'VRS_' besteht gefolgt vom technischen Namen des Mandanten. Hierzu können Sie folgendes Kommando verwenden (angepasst an Umgebung und Ziel der Installation):

    - **Windows:** MaintenanceTool.cmd --dumpTenantSchemaDDL --tenantSchemaName <tenantSchemaName> --sqlDumpFile <dumpFileName> --executeSql --dbBrand <dbBrand> -dbDriverClass <driverClassName> -jdbcUrl <jdbcUrl> -dbUser <adminUserName> -dbPassword <adminUserPassword>

    - **Linux/Solaris:** MaintenanceTool.sh --dumpTenantSchemaDDL --tenantSchemaName <tenantSchemaName> --sqlDumpFile <dumpFileName> --executeSql --dbBrand <dbBrand> -dbDriverClass <driverClassName> -jdbcUrl <jdbcUrl> -dbUser <adminUserName> -dbPassword <adminUserPassword>

    Das neue Mandantenschema wird registriert werden, wenn ein Execution Server Benutzer sich das erste Mal für den Mandanten einlogt. Damit diese Registrierung funktioniert, muss der Execution Server Datenbankbenutzer in der Lage sein, existierende Schemata auszuwählen, um zu verifizieren, dass das Datenbankschema für den Mandanten tatsächlich existiert.

    Wenn Sie das Präfix 'VRS_' nicht für das Mandantenschema verwenden wollen, oder einfach einen beliebigen Namen für das Mandantenschema verwenden wollen, dann wird der Execution Server nicht in der Lage sein, die automatische Registrierung für dieses Mandantenschema durchzuführen. Daher müssen Sie in diesem Fall die Registrierung mithilfe des Execution Server Maintenance Tools manuell durchführen, was auch als 'anbinden' bezeichnet wird.

2. Legen Sie die Datenbanktabellen für einen einzelnen Mandanten an, indem Sie folgendes Kommando ausführen (angepasst an Umgebung und Ziel der Installation): .

    - **Windows:** MaintenanceTool.cmd --dumpExecutionServerDDL --tenantSchemaName <tenantSchemaName> --tablePrefix <tablePrefix> --executeSql --sqlDumpFile <dumpFileName> --dbBrand <dbBrand> -dbDriverClass <driverClassName> -jdbcUrl <jdbcUrl> -dbUser <adminUserName> -dbPassword <adminUserPassword>

    - **Linux/Solaris:** MaintenanceTool.sh --dumpExecutionServerDDL --tenantSchemaName <tenantSchemaName> --tablePrefix <tablePrefix> --executeSql --sqlDumpFile <dumpFileName> --dbBrand <dbBrand> -dbDriverClass <driverClassName> -jdbcUrl <jdbcUrl> -dbUser <adminUserName> -dbPassword <adminUserPassword>

    Die 'tenantId' bezieht sich auf die technische ID des Mandanten, so wie sie in dem Identity Management geführt wird, an das der Execution Server angebunden ist.

3. Binden Sie, falls nötig, das Mandantenschema an einen Mandanten, indem Sie folgendes Kommando ausführen (angepasst an Umgebung und Ziel der Installation):

    - **Windows:** MaintenanceTool.cmd --dumpTenantBindingSQL --tenantId <tenantId> --globalSchemaName <globalSchemaName> --tenantSchemaName <tenantSchemaName> --executeSql --sqlDumpFile <dumpFileName> --dbBrand <dbBrand> -dbDriverClass <driverClassName> -jdbcUrl <jdbcUrl> -dbUser <adminUserName> -dbPassword <adminUserName>

    - **Linux/Solaris:** MaintenanceTool.sh --dumpTenantBindingSQL --tenantId <tenantId> --globalSchemaName <globalSchemaName> --tenantSchemaName <tenantSchemaName> --executeSql --sqlDumpFile <dumpFileName> --dbBrand <dbBrand> -dbDriverClass <driverClassName> -jdbcUrl <jdbcUrl> -dbUser <adminUserName> -dbPassword <adminUserPassword>

4. Geben Sie dem Datenbankbenutzer, der vom Execution Server für die Datenbankverbindung verwendet wird, CRUD-Berechtigungen auf den erzeugten Execution Server Datenbankobjekten.

```
-- Take the result of this select statement and execute it, to grant the necessary privileges
select grant_statements||owner||'.'||db_obj.object_name||' to <username>;' as grant_statements
from (
select 'grant select on ' as grant_statements, v.owner, v.VIEW_NAME as object_name
from sys.all_views v
union all
select 'grant select,insert,update,delete on ',
t.owner, t.table_name
from sys.all_tables t
) db_obj
where db_obj.owner = '<tenantSchemaName>' and
db_obj.object_name like '<tablePrefix>%'
```

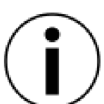
Beispiel 2.4. Manuelles Aufsetzen der Mandanten: Berechtigungen

2.6. Installation der Lizenz

Der Execution Server benötigt eine gültige Lizenzdatei, da ansonsten keinerlei Anfragen bearbeiten werden. Mit der Lizenz Verwaltung auf der Sicht **Wartung**, können Lizenzen hinzugefügt werden. Es gibt zwei Möglichkeiten, wie der Server eine Lizenzdatei finden kann:

1. Die Lizenzdatei liegt im Ordner `.visualrules6` im Benutzerverzeichnis des Benutzers der den Server ausführt. Dort wird standardmäßig nach Lizenzdateien gesucht.

2. Es kann ein Dateipfad zur Lizenzdatei mit der Einstellung `license.file` angegeben werden. Dies wird nur benötigt, wenn sich keine Lizenzdatei im Standardverzeichnis befindet. Lesen Sie Abschnitt 2.7, „Konfiguration des Execution Servers“ , um mehr Details darüber zu erhalten, wie dies bewerkstelligt werden kann.



Im Fall einer ungültigen oder fehlenden Lizenz wird nach der Anmeldung automatisch zur Lizenz Verwaltung weitergeleitet. Ebenso wird eine Fehlermeldung in das Log geschrieben. Dort finden sich ebenfalls Informationen zu den Stellen, an denen der Execution Server nach Lizenzen sucht. Dafür muss Logging eingeschalten und auch für den Level INFO konfiguriert sein.

2.7. Konfiguration des Execution Servers

Nach erfolgreicher Installation des Execution Servers ist keine weitere Konfiguration notwendig. Dieses Kapitel beschreibt wie Sie den Execution Server konfigurieren, falls Sie die Konfiguration zu einem späteren Zeitpukt ändern wollen oder Einstellungen vornehmen wollen, die nicht im Installations-Wizard möglich sind (z.B. für spezielle Anforderungen in Produktionsumgebungen).

2.7.1. Execution Server Home

Das Execution Server Home ist ein Ordner der für die Speicherung von Daten und Konfigurationen des Execution Server benutzt wird. Der Ordnername entspricht normalerweise dem Kontext-Root den Sie beim Deployment der Execution Server Anwendung gewählt haben. Er befindet sich in einem Verzeichnis `visualrules` im Benutzerverzeichnis des Betriebssystembenutzers, der den Execution Server startet. Bei Bedarf wird der Ordner automatisch angelegt.

Es ist ebenfalls möglich, ein anderes Verzeichnis als `/<user.home>/visualrules/` zu verwenden durch das Setzen einer System Property für die Java VM, welche den Applikations Server betreibt. Der Schlüssel ist `visualrules.executionserver.home` und der Wert ist ein gülter Dateipfad der auf ein schreib- und lesbares Verzeichnis zeigt. Beispielsweise könnte das folgendermaßen aussehen, wenn der Wert per Kommandozeile angegeben wird: `-Dvisualrules.executionserver.home=D:\my_es_home`

Tabelle 2.1. Execution Server Home Inhalt

config/executionserver.properties	Die Execution Server Konfigurationsdatei
config/vr_logging.config	Konfigurationsdatei für die Laufzeit Protokollierung
logs	Ordner der die Protokoll Dateien des Execution Servers enthält

2.7.2. Execution Server Konfiguration

Der Execution Server wird durch den Installationsassistenten installiert und konfiguriert. Alle Einstellungen werden in einer Datei im Execution Server Home Verzeichnis persistiert. Es ist möglich, diese Werte zu editieren für den Fall, dass nach der Installation eine Änderung notwendig ist. Hat sich beispielsweise eine URL für das Identity Management (IM) geändert, so kann dies in der Konfigurationsdatei geändert werden, ohne die Anwendung erneut zu installieren. Änderungen an der Konfigurationsdatei dürfen nicht vorgenommen werden, wenn der Execution Server in Betrieb ist.



Bei Eingabe von Werten in eine Properties Datei, müssen unter Umständen bestimmte Zeichen maskiert werden.

Unten aufgelistet finden Sie die möglichen Optionen zur Konfiguration des Execution Servers.

Die typischen Konfigurationseinstellung, wie sie vom Installationsassistenten erstellt werden, sind:

`executionserver.jndi.name`
: Der JNDI Resource Name der zu verwendenden Datenbank.

`artifactstorage.db.brand`
: Der Name der verwendeten Datenbank. Gültige Werte sind: `H2`, `MYSQL`, `ORACLE`, `MSSQL` (Groß-/Kleinschreibung wird nicht beachtet)

`artifactstorage.custom.prefix`
: Der benutzer-definierte Tabellenpräfix, der bei der Trennung der Anwendungsdaten Verwendung findet. Der Wert muss zwischen ein und drei Zeichen lang sein, mit einem Buchstaben anfangen und aus Zahlen oder einfachen Buchstaben bestehen. Etwas formeller, es muss dem folgenden regulären Ausdruck Genüge getan sein: `[A-Z][0-9A-Z]{0,2}`

`mts.schema.name`
: Der Wert ist der Name des Datenbankschemas, welches für die Verwaltung von verschiedenen Anwendungen und Mandanten verwendet wird, wie in Abschnitt 1.4.2.1, „Trennung der Mandantendaten“ beschrieben.

`im.application.id`
: Die eindeutige ID (uuid) mit der diese Anwendung beim Identity Management (IM) registriert ist. Zu beachten ist, dass dies **nicht** der Name der Anwendung ist und der Wert nur von IM geliefert werden kann.

`im.frontend.url`
: Die URL der Webkonsolen Anwendung von Identity Management.

`im.backend.url`
: Die URL der Backend Anwendung von Identity Management.

Es gibt weitere, wenn auch optionale Konfigurationseinstellungen, welche nicht vom Installationsassistenten gesetzt werden:

`license.file`
: Gültiger Dateipfad zu der Lizenzdatei des Execution Servers. Wird nur benötigt, wenn sich die Lizenz nicht im Standardverzeichnis befindet.. Es ist ebenfalls möglich, eine Lizenz in der Lizenz Verwaltung in der Sicht **Wartung** hinzuzufügen.

`localstorage.workingdir`
: Ort, wo der Execution Server Artefakte als Jars speichert. Standardmäßig wird dafür das temporäre Verzeichnis des Web Containers benutzt, welches durch das Kontextattribut

`javax.servlet.context.tempdir` bestimmt ist. Der Wert muss ein Pfad zu einem les- und beschreibbaren Ordner sein.

`metadata.custom.mapper`
Der voll qualifizierte Klassenname eines benutzer-definierten Metadata Mappers. Die Benutzung eines benutzerdefinierten Metadata Mappers ist in Abschnitt 6.2.1.3, „Benutzerdefinierter Metadata Mapper“ genauer beschrieben.

2.7.3. Laufzeitprotokollierung

Die Laufzeitprotokollierung wird vom Execution Server automatisch eingerichtet. Beim Startvorgang wird die Laufzeitprotokollierung initialisiert und Nachrichten werden in eine Protokolldatei (auch log genannt) geschrieben, welche sich im `logs` Ordner im Execution Server Home befindet. Eine Protokoll Datei wird bis zu 3 Megabyte (MB) groß und wird dann archiviert. Es gibt maximal drei Archive die eine Nummer im Dateinamen tragen. Zum Beispiel ist "3" das dritte und somit älteste Archiv. Der Execution Server schreibt zwei unterschiedliche Protokolle:

- `executionserver.log` enthält die Nachrichten vom Execution Server und verwendeter Komponenten von Drittanbietern
- `performance.log` wird für die Analyse der Leistungsfähigkeit verwendet. Die Datei wird automatisch angelegt. Die Datei ist leer, da die Analyse standardmäßig abgeschalten ist.



Die Ausgabe von Nachrichten mit Aktionen des Typs "Log Eintrag schreiben" ist weitestgehend deaktiviert, um die Ausgabe von mandatenspezifischen Informationen zu verhindern. Wird jedoch die **Kategorie** Eigenschaft von Aktionen vom Typ *"Log Eintrag schreiben"* verwendet, so werden die Nachrichten in die `executionserver.log` Datei geschrieben.

Die Protokoll Datei kann in der Webkonsole eingesehen werden, wo es auch möglich ist, Einstellungen für die Laufzeitprotokollierung vorzunehmen. Mehr Informationen dazu finden sich in Abschnitt 4.5.2, „Konfiguration und Anzeige von Nachrichten der Laufzeitprotokollierung“.

Sollte der Fall eintreten, dass der Execution Server nicht starten kann oder es aus anderen Gründen nicht möglich ist, auf die Webkonsole zuzugreifen, dann kann die Laufzeitprotokollierung in der `vr_logging.config` Datei im Execution Server Home konfiguriert werden.



Das sollte nur in Ausnahmesituationen benutzt werden und ist nicht für die Verwendung im Normalbetrieb gedacht. Diese Einstellung ist nur möglich, wenn der Execution Server nicht läuft. Die Konfigurationsdatei wird nur beim Startvorgang eingelesen und Einstellungen in der Webkonsole überschreiben Werte in der Datei.

Bevor die Protokollierung in der Konfigurations Datei vorgenommen werden kann, muss sichergestellt sein, dass der Execution Server nicht läuft. Dann kann die `vr_logging.config` Datei in einem Texteditor geöffnet werden. Die Datei enthält XML Elemente names `loggers` und `logger`. Diese haben Attribute names `level` welche den Protokollierungsgrad (log level) spezifizieren, die ähnlich zu den in Simple Logging Facade for Java (SLF4J) sind. Der Wert kann geändert werden zu: `TRACE`, `DEBUG`, `INFO`, `WARN` oder `ERROR`. Für die Fehlersuche eignet sich `DEBUG`. Nach dem Speichern der Änderung, sollte die Protokolldatei geleert werden, damit Fehler leichter zu finden sind. Beim anschließenden Start des Execution Servers wird die neue Konfiguration angewendet und Nachrichten mit dem eingestellten Protokollierungsgrad sollten in der Protokolldatei zu finden sein.

Kapitel 3. Erstellung und Bereitstellung von Rule Services

3.1. Konzepte

3.1.1. Rule Service

Der Visual Rules Execution Server ist eine Ausführungsumgebung für **Regeln**. Regeln werden mit dem Visual Rules Modeler erstellt und können dann auf dem Visual Rules Execution Server bereitgestellt werden. Einmal bereitgestellt, stehen die Regeln für jegliche andere Applikationen oder Systeme, welche die Regeln aufrufen möchten, zur Verfügung. Der Visual Rules Execution Server stellt eine Web Service Schnittstelle zur Verfügung, um Regeln aufzurufen. Auf diese Weise wird jede Regel zu einem separaten Service - ein sogenannter Rule Service.

Die Regeln für einen Rule Service müssen für die Anwendung in eine Regelbibliothek gepackt werden. Diese Bibliothek und die entsprechenden Services haben Versionsnummern, um eindeutig während des Lebenszyklusses eines Regelprojektes identifiziert werden zu können. Mehrere Versionen des gleichen Rule Service können im Visual Rules Execution Server bereitgestellt und zur gleichen Zeit ausgeführt werden.

Ein Rule Service wird durch eine WSDL Datei (Web Service Description Language) beschrieben, die automatisch durch Visual Rules während der Erstellung der Regelbibliothek generiert wird, um dann nach der Bereitstellung verfügbar zu sein. Auf die WSDL Dateien aller eingesetzten Rule Services kann über die Execution Server Webkonsole zugegriffen werden.

Weiterführende Konzepte.

- Abschnitt 3.1.2, „Regelbibliothek“
- Abschnitt 3.1.3, „Versionen von Regelbibliothek und Rule Service“

Weiterführende Arbeitsschritte.

- Abschnitt 3.2.1, „Regeln festlegen, die als Rule Service exportiert werden sollen“
- Abschnitt 4.2.5, „Anzeige der WSDL-Datei eines Rule Service“

3.1.2. Regelbibliothek

Um einen Rule Service im Visual Rules Execution Server zu verwenden, muss das entsprechende Regelprojekt in eine sogenannte Regelbibliothek gepackt werden. Regelbibliotheken sind JAR Dateien, die den generierten Regelcode für alle Regelmodelle, die im Regelprojekt enthalten sind, beinhalten (siehe Abbildung 3.1, „Regelprojekt, Regelbibliothek und Rule Service“).

Die Regelbibliothek beinhaltet somit die WSDL Datei (und XML Schemas) für das Regelmodell, das als Web Service exportiert wird (es kann nur ein Regelmodell eines Regelprojekts als Rule Service exportiert werden, die anderen Regelmodelle bleiben intern). Die WSDL und XML Schema Dateien werden automatisch von Visual Rules erstellt, wenn es eine Regelbibliothek paketiert.

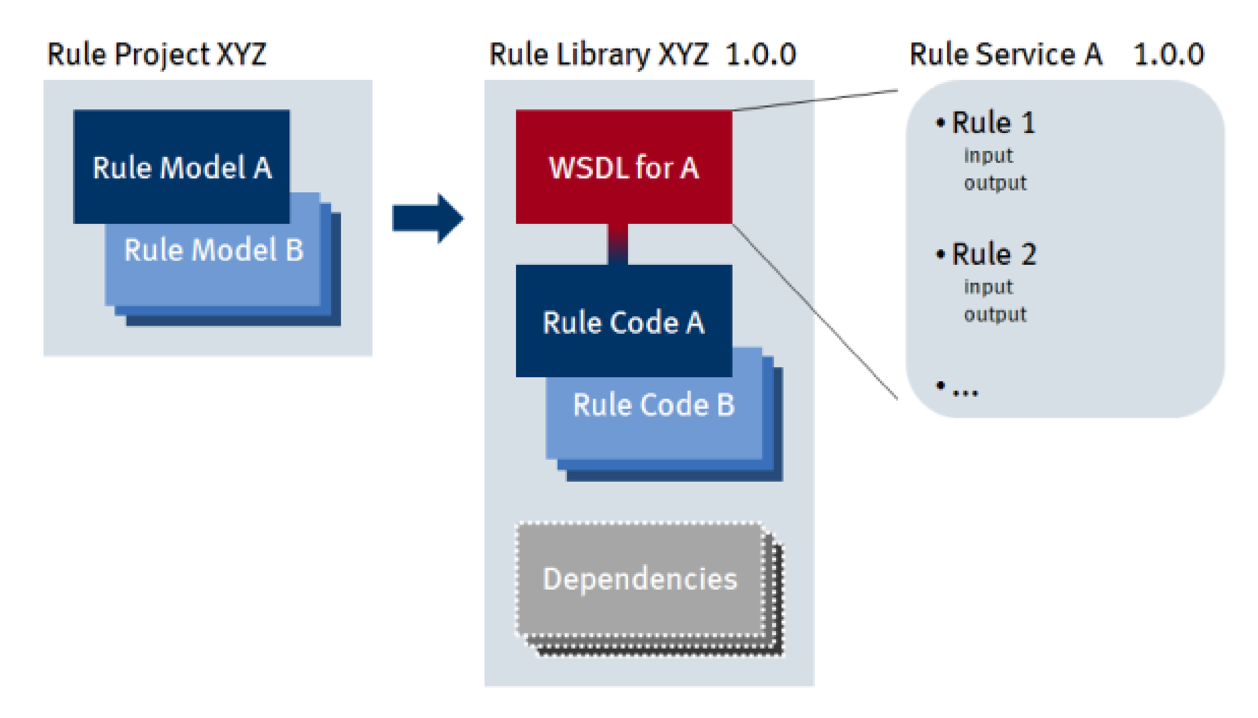


Abbildung 3.1. Regelprojekt, Regelbibliothek und Rule Service



Der Visual Rules Modeler paketiert Regelbibliotheken, so dass sie den Regelcode von allen Regelmodellen im Projekt UND den Regelcode von allen abhängigen Regelmodellen enthalten. Die Regelbiblothek beinhält auch alle Klassen von anderen Bibliotheken und Java Projekten auf dem Erstellungspfad des Projekts. Auf diese Weise enthält die resultierende Regelbibliothek alle benötigten Klassen für die Regeln zur Ausführung. Beachten Sie jedoch, dass die Visual Rules Laufzeit Bibliotheken nicht in die Regelbibliothek gepackt werden. Sie werden zur Laufzeit automatisch zum Klassenpfad des Execution Servers hinzugefügt.

Weiterführende Konzepte.

- Abschnitt 3.1.1, „Rule Service“
- Abschnitt 3.1.3, „Versionen von Regelbibliothek und Rule Service“

Weiterführende Arbeitsschritte.

- Abschnitt 3.2.1, „Regeln festlegen, die als Rule Service exportiert werden sollen“

3.1.3. Versionen von Regelbibliothek und Rule Service

Eine Regelbibliothek hat immer eine Versionsnummer. Diese Versionsnummer wird verwendet, um eine spezifische Regelbibliothek und Rule Service während der Bereitstellung und der Ausführung von Regeln zu identifizieren.

Die Versionsnummer besteht aus drei individuellen Nummern, die durch Punkte getrennt sind, d.h. `1.0.0` oder `12.4.11`. Versionsnummern werden verwendet, um eindeutig eine spezifische Version einer Regelbibliothek zu identifizieren. Dies ist notwendig, weil während des Lebenszyklusses eines Regelprojekts viele verschiedene Versionen für die Ausführungsumgebung zur Verfügung gestellt werden können. So können mehrere Versionen des gleichen Regelprojekts zur gleichen Zeit eingesetzt werden. Durch Verwendung der Versionsnummer, kann ein Aufrufer genau spezifizieren, welche Version der Regel aufgerufen werden soll.

Die drei Komponenten einer Versionsnummer werden "major", "minor" und "micro" genannt. Höhere Nummern werden als "neuere" Versionen angesehen. Die micro und/oder minor Komponenten können weggelassen werden und werden dann als `0` behandelt, beispielsweise entspricht die Versionsnummer `2.6` der Nummer `2.6.0`, und `1` entspricht `1.0` beziehungsweise `1.0.0`.

Optional kann nach der "micro" Komponente, eine weitere Komponente angegeben werden, die "qualifier" genannt wird. Der Qualifier wird durch einen Bindestrich (-) getrennt und kann zusätzlich zur Identifikation verwendet werden, beispielsweise kann dies ein Zeitstempel sein. Der Qualifier kann nicht nur Nummern enthalten, sondern

auch Buchstaben (`A-Z`, `a-z`), Unterstriche (_) und Bindestriche (-). Qualifier werden lexikographisch geordnet, um zu bestimmen, welche Version "neuer" ist.

Unter Umständen kann es dazu kommen, dass zwei Rule Services die gleiche Version haben, aber nur einer der beiden einen "qualifier" hat. Wenn in so einem Fall der Rule Service ohne eine spezifizierte Version aufgerufen wird, dann wird der Rule Service ohne "qualifier" als "neuer" angesehen.

Weiterführende Konzepte.

- Abschnitt 3.1.1, „Rule Service“
- Abschnitt 3.1.2, „Regelbibliothek“

Weiterführende Arbeitsschritte.

- Abschnitt 3.2.4, „Bereitstellen von Regelprojekten vom Visual Rules Modeler“

3.1.4. Visual Rules Archiv

Der Execution Server unterstützt auch das Visual Rules Archiv, welches neben dem Rule Service auch noch alle weiteren Bibliotheken, die zur Ausführung eines Rule Services notwendig sind, und eine Abhängigkeitsbeschreibung enthält.

In dieser Form können Rule Services von einem Visual Rules Execution Server herunter- bzw. hochgeladen werden (z.B. um Rule Services von einer Test- auf eine Produktionsumgebung zu übertragen).

Die zugehörige Archivdatei besitzt die Endung VRA.

Weiterführende Konzepte.

- Abschnitt 3.1.1, „Rule Service“
- Abschnitt 3.1.2, „Regelbibliothek“

Weiterführende Arbeitsschritte.

- Abschnitt 4.2.6, „Herunterladen eines Rule Service“
- Abschnitt 4.2.3, „Hinzufügen eines Rule Service mittels Visual Rules Archiv“

3.1.5. Rule Service Einstellungen

Rule Services können zusätzliche Einstellungen haben, die im folgenden beschrieben sind.

3.1.5.1. Aktiv

Ein Rule Service ist standardmäßig aktiv und kann somit aufgerufen werden. Es kann allerdings auch Situationen geben, in denen Rule Services zwar bereitgestellt sind, aber nicht aufgerufen werden sollen. Für solche Zwecke kann ein Rule Service deaktiviert werden, wodurch dieser nicht mehr aufgerufen werden kann. Für Aufrufer ist es dann nicht unterscheidbar, ob der Rule Service gelöscht wurde oder nur deaktiviert wurde.

3.1.5.2. Gültig von und Gültig bis

Diese Werte werden vom Standard-Metadata Mapper in einer generischen Rule Service Anfrage verwendet. Beim Einsatz eines benutzerdefinierten Metadata Mappers, wie in Abschnitt 6.2.1.3, „Benutzerdefinierter Metadata Mapper“ beschrieben, kann das ebenfalls genutzt werden.

3.1.5.3. Name der aktiven Konfiguration (Active Configuration Name)

Im Visual Rules Modeler können mehrere Konfigurationen für die Regelausführung auf einem Regelmodell definiert werden. Diese können beispielsweise in der *Visual Rules Rule Execution API* verwendet werden, um Implementierungen von Aktionen auszutauschen. Mehr Informationen zu diesem Thema finden sich im *Java Integration Handbuch*.

Der Execution Server bietet Aufrufern ebenfalls die Möglichkeit, den Namen der zu verwendenden Konfiguration anzugeben. Im Englischen und den technischen Schnittstellen wird hierfür der Begriff `active configuration name` verwendet. Zu beachten ist hierbei, dass diese Angabe im Aufruf in älteren Versionen der *Visual Rules Rule Execution API* und im Execution Server als `binding` bezeichnet wird.

Jeder Rule Service kann konfiguriert werden, einen speziellen Namen als Standardwert für die aktive Konfiguration zu verwenden. Dieser kann anders sein, als derjenige, der normalerweise in der *Visual Rules Rule Execution API* verwendet wird. Dies wird dann als `active configuration name` verwendet, wenn ein Rule Service ausgeführt wird. Ein Aufrufer kann ein `active configuration name` auch in der Rule Service Anfrage angeben (für Details siehe Abschnitt 5.1.2, „Format der Rule Service Anfrage"). Die Einstellung in der Anfrage übertrumpft in diesem Fall immer den Wert der am Rule Service eingestellt wurde.

3.1.5.4. Statistiklevel

Der **Statistiklevel** erlaubt es, den Detaillierungsgrad der aufgezeichneten Statistiken einzustellen. Das ist eine Eigenschaft der *Visual Rules Rule Execution API*, die detailliert im *Java Integration Handbuch* beschrieben wird.

Der Execution Server sammelt Informationen von Ausführungen von Rule Services. Im Rahmen dieser Funktionalität können auch Statistiken von Regeln aufgezeichnet werden. Das kann auf zwei Arten verwendet werden. Zum einen kann der Statistiklevel pro Rule Service eingestellt werden, welcher dann als Standardwert bei der Ausführung verwendet wird. Zum anderen können Aufrufer einen Statistiklevel auch in der Rule Service Anfrage spezifizieren (siehe Abschnitt 5.1.2, „Format der Rule Service Anfrage"), welcher dann statt der Einstellung des Rule Service verwendet wird.

Generell gilt: je höher der Detaillierungsgrad, desto mehr Information wird aufgezeichnet. Die folgenden Detaillierungsgrade sind verfügbar:

- **Quiet**: Statistiken der Regeln werden nicht aufgezeichnet und sind nicht zum Herunterladen verfügbar.
- **Low**: Besuche von Regelelemente werden gezählt.
- **Medium**: Zeit, die in einem Regelelement verbracht wird, wird gemessen.
- **High**: Minimale und maximale Zeit, die in einem Regelelement verbracht wird, wird gemessen.

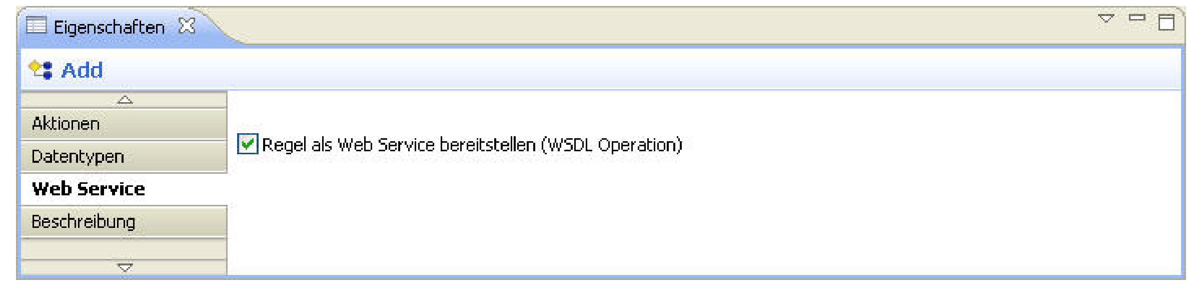
Zusätzlich zu den Leveln in der *Visual Rules Rule Execution API*, hat der Execution Server den Level **Quiet**, mit dem die Aufzeichnung der Statistik abgeschalten werden kann. Durch die Einstellung **Switched off**, kann ein Rule Service so konfiguriert werden, dass keine Ausführungen aufgezeichnet wird. Ein spezifizierter Statistiklevel eines Aufrufers hat somit keine Auswirkung.

3.2. Arbeitsschritte

3.2.1. Regeln festlegen, die als Rule Service exportiert werden sollen

Nur exportierte Regeln können durch einen Rule Service Client aufgerufen werden. Somit muss mindestens eine Regel eines Regelmodells exportiert werden, um einen Rule Service zu erstellen. Um zu spezifizieren, welche Regeln exportiert werden sollen, führen Sie folgendes durch:

1. Im **Projekt Explorer** oder **Regel Explorer** wählen Sie eine Regel zum Exportieren aus.
2. Gehen Sie auf die Seite **Web Service** in der Sicht **Eigenschaften**.
3. Aktivieren Sie das Kästchen **Regel als Web Service bereitstellen (WSDL operation)**.



Weiterführende Konzepte.

- Abschnitt 3.1.1, „Rule Service“
- Abschnitt 3.1.2, „Regelbibliothek“

3.2.2. XML Namespace für Rule Services definieren

Die generierte WSDL und die XML Schemata für einen Rule Service verwenden einen spezifischen XML Namespace. Wenn Sie die Namespace URI für einen Rule Service anpassen möchten, führen Sie folgende Schritte durch:

1. Im **Projekt Explorer** oder **Regel Explorer** wählen Sie das entsprechende Regelmodell aus. Dies ist das Regelmodell, welches für den Rule Service exportiert wird.

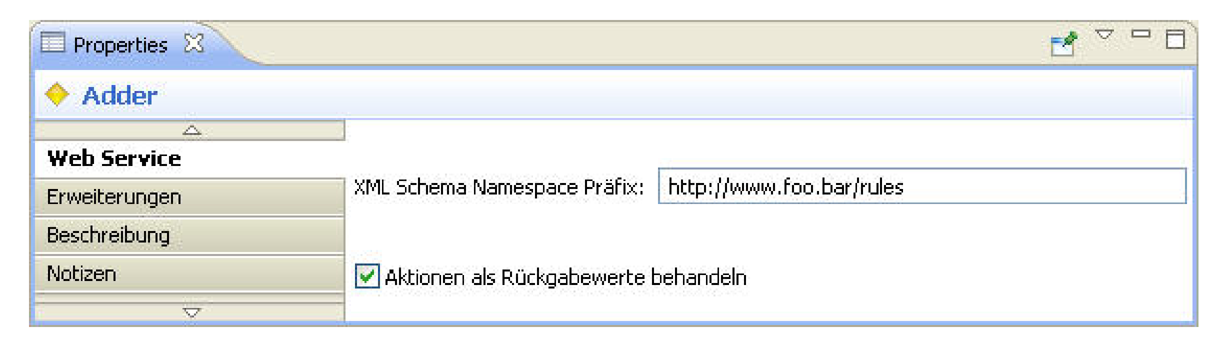
2. Gehen Sie auf die Seite **Web Service** in der Sicht **Eigenschaften**.

3. Geben Sie den **XML Schema Namespace Präfix** ein. Dies muss eine gültige URI sein (typischerweise wird http:// als Schema verwendet).

   Der vollständige Namespace URI im WSDL und XML Schema wird später aus diesem Präfix gebildet, gefolgt vom Segment `/vrpath`, gefolgt vom Pfad der spezifischen Regel.



Ist kein Präfix angegeben, wird `http://www.visual-rules.com` als Standardwert verwendet.



Weiterführende Konzepte.

- Abschnitt 3.1.1, „Rule Service“
- Abschnitt 3.1.2, „Regelbibliothek“
- Abschnitt 5.1.2, „Format der Rule Service Anfrage“
- Abschnitt 5.1.3, „Format der Rule Service Antwort“

3.2.3. Einstellen von Aktionen als Rückgabewert

Aktionen sind ein Teil des Ergebnisses eines Regelaufrufs. Somit sind sie auch Teil der Rückgabe der generierten XML Schema Dateien für einen Rule Service. Es ist möglich einzustellen, ob Aktionen als Rückgabewert behandelt werden. Das führt dann zu unterschiedlichen XML Schema Definitionen für einen Web Service. Um dies einzustellen, führen Sie folgende Schritte durch:

1. Im **Projekt Explorer** oder **Regel Explorer** wählen Sie das entsprechende Regelmodell aus. Dies ist das Regelmodell, welches für den Rule Service exportiert wird.

2. Gehen Sie auf die Seite **Web Service** in der Sicht **Eigenschaften**.

3. Durch Einstellen der Checkbox kann das Verhalten umgestellt werden. Das hat einen Effekt auf die Elemente, welche in der Antwort zurückgeliefert werden.

3.2.4. Bereitstellen von Regelprojekten vom Visual Rules Modeler

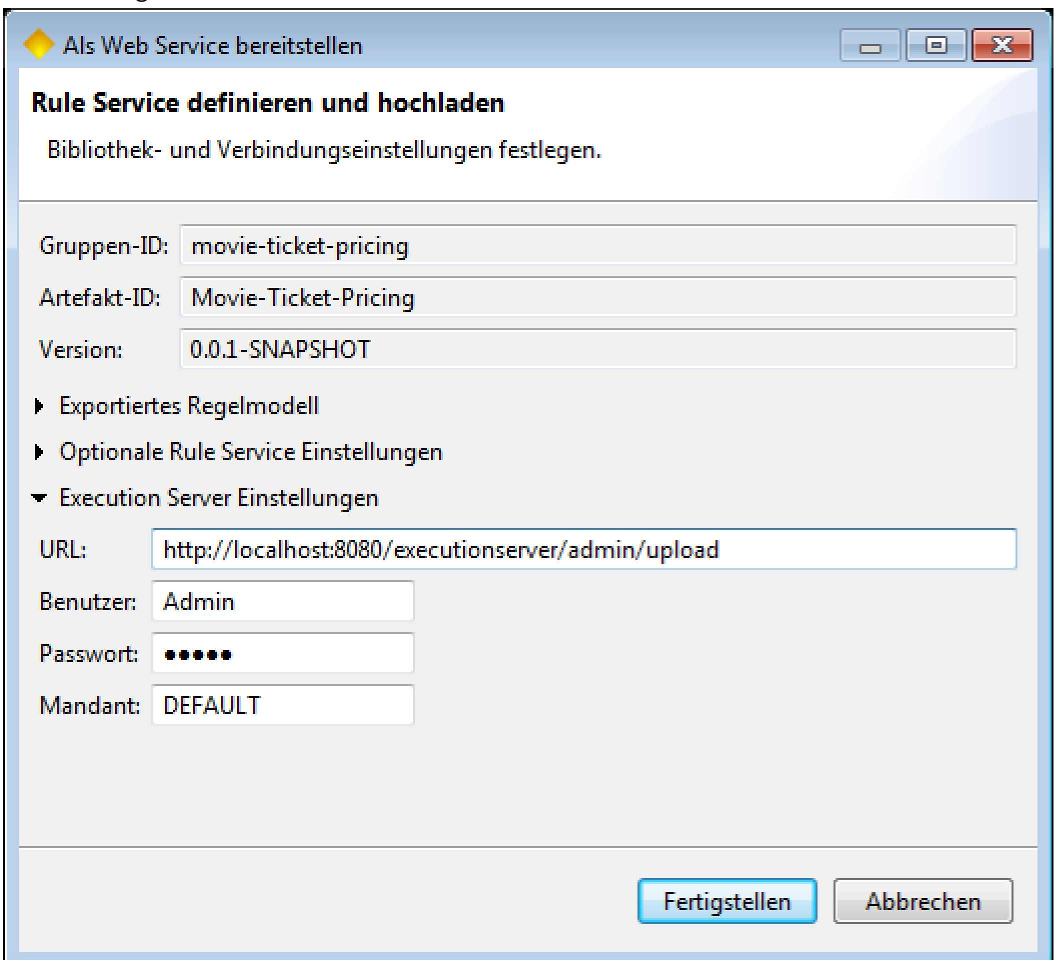
Regelprojekte können direkt im *Visual Rules Modeler* paketiert und auf dem Execution Server bereitgestellt werden. Diese Funktionalität ist auch in *Visual Rules Builder* und *Visual Rules Team Platform* (beide separat verfügbar) enthalten.



Für die Bereitstellung muss der Benutzer über die erforderliche Berechtigung verfügen. Siehe auch Abschnitt 1.3, „Berechtigungs-Konzept“.

1. Im **Regel Explorer** oder **Projekt Explorer** machen Sie einen Rechtsklick auf das Regelprojekt, das bereitgestellt werden soll.

2. Wählen Sie den Visual Rules > Als Web Service bereitstellen Menüeintrag. Der Assistent für die Bereitstellung erscheint.



3. Die **Artefakt-ID**, **Gruppen-ID** und **Version** sind auf der `ruleproject.vr` definiert und werden zur Identifizierung des Artefakts (JAR) verwendet. Diese **Version** gilt für die Regelbibliothek selbst und den Rule Service, der bereitgestellt wird.

4. Wenn das Regelprojekt mehrere Regelmodelle enthält, müssen Sie auswählen, welches Regelmodell exportiert werden soll. Erweitern Sie den Abschnitt **Exportiertes Regelmodell** und wählen Sie das Regelmodell zum Exportieren aus. Dies ist nicht notwendig, wenn das Regelprojekt nur ein einziges Regelmodell hat.

5. Im Abschnitt **Optionale Rule Service Einstellungen** können Werte für die Gültigkeitsdauer eingestellt und der Rule Service aktiviert bzw. deaktiviert werden.

6. Erweitern Sie den Abschnitt **Execution Server Einstellungen** und prüfen Sie die Werte für die Execution Server **URL**, den **Benutzer**, das **Passwort** und den **Mandant**.



Die URL endet immer mit `/admin/upload`. Der Host Name und der Kontext Name hängen von Ihrer Installation des Execution Server ab.

In der Abbildung oben läuft der Execution Server auf dem lokalen Rechner mit dem Port 8080 und der Webapplikations-Kontext wurde `executionserver` genannt.

7. Klicken Sie auf **Fertig stellen**.
8. Visual Rules Modeler wird nun die Abhängigkeiten des Regelprojekts analysieren, den Regelcode generieren und diesen in eine Regelbibliothek mit allen anderen Abhängigkeiten auf dem Erstellungspfad der involvierten Projekte packen. Die Regelbibliothek wird dann zum Execution Server hochgeladen.

Weiterführende Konzepte.

- Abschnitt 3.1.1, „Rule Service"
- Abschnitt 3.1.2, „Regelbibliothek"

Weiterführende Arbeitsschritte.

- Abschnitt 3.2.1, „Regeln festlegen, die als Rule Service exportiert werden sollen"

Kapitel 4. Arbeiten mit der Webkonsole

Die folgenden Abschnitte enthalten Beschreibungen verschiedener Aufgaben, die mit der Webkonsole ausgeführt werden können, z.B. bereitgestellte Rule Services verwalten.

4.1. Aufruf der Webkonsole

Nach erfolgreicher Installation des Execution Servers kann die Webkonsole in einem Web Browser aufgerufen werden. Sie ist unter der Adresse http://<server>:<port>/<context-name> erreichbar, wobei die Werte meist durch den Application Server vorgegeben sind.

Bei einer Bereitstellung auf einem Apache Tomcat, unter Verwendung dessen Standardkonfiguration, lautet die Adresse beispielsweise http://localhost:8080/executionserver-6.x.y (wobei x und y Platzhalter für die Versionsnummern aus dem Dateinamen der WAR Datei des Execution Servers darstellen).

Es erscheint die folgende Startseite zur Anmeldung:



Wenn beim Start des Execution Servers keine gültige Lizenz zur Verfügung stand, erscheint die Startseite mit einer entsprechenden Fehlermeldung.

Weiterführende Aufgaben.

- Abschnitt 2.6, „Installation der Lizenz“

4.1.1. Einstellung einer bestimmten Sprache

Die Seiten der Webkonsole werden in der Sprache angezeigt, die Sie als bevorzugte Sprache in Ihrem Web Browser eingestellt haben. Sollte diese Sprache von der Webkonsole noch nicht unterstützt werden, dann werden die Seiten in *Englisch* angezeigt.

Es ist möglich, explizit die *deutsche* bzw. *englische* Sprache für die Webkonsole einzustellen:

1. Öffnen Sie eine Seite der Webkonsole in Ihrem Web Browser.
2. Bewegen Sie die Maus über das Flaggensymbol im rechten Teil der Menüleiste.
3. Klicken Sie im Kontextmenü auf die Flagge , wenn Sie die *deutsche* Sprache einstellen wollen bzw. auf die Flagge , wenn Sie die *englische* Sprache wünschen.

4.2. Verwaltung bereitgestellter Rule Services

In der Sicht *Rule Services* des Visual Rules Execution Servers können Sie bereitgestellte Rule Services verwalten.

Wählen Sie in der Menüleiste der Webkonsole den Eintrag Rule Services, um diese Sicht zu öffnen.

Hier haben Sie folgende Möglichkeiten:

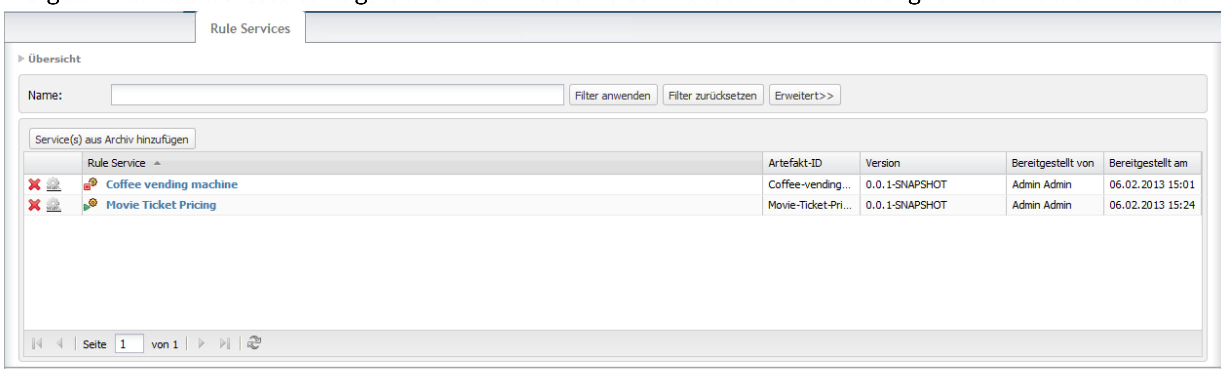
- Abschnitt 4.2.1, „Anzeige bereitgestellter Rule Services“

- Abschnitt 4.2.2, „Filterung angezeigter Rule Services“
- Abschnitt 4.2.3, „Hinzufügen eines Rule Service mittels Visual Rules Archiv“
- Abschnitt 4.2.4, „Löschen eines bereitgestellten Rule Service“
- Abschnitt 4.2.5, „Anzeige der WSDL-Datei eines Rule Service“
- Abschnitt 4.2.6, „Herunterladen eines Rule Service“
- Abschnitt 4.2.7, „Anzeige der Eigenschaften und Änderung der Einstellungen eines Rule Service“
- Abschnitt 4.2.8, „Verwaltung der Metadaten eines Rule Service“
- Abschnitt 4.2.9, „Anzeige der Ausführungen eines Rule Service“
- Abschnitt 4.2.10, „Anzeige der erforderlichen Bibliotheken eines Rule Service“

4.2.1. Anzeige bereitgestellter Rule Services

Wählen Sie in der Menüleiste der Webkonsole den Eintrag Rule Services aus, um die Sicht *Rule Services* zu öffnen.

Die geöffnete Übersichtsseite zeigt alle auf dem Visual Rules Execution Server bereitgestellten Rule Services an:



Standardmäßig werden folgende Eigenschaften der Rule Services angezeigt:

- Aktiviert oder deaktiviert
- Version
- Artefakt-ID
- Bereitgestellt von
- Bereitgestellt am

Sie können das Layout der angezeigten Eigenschaften beeinflussen, indem Sie beispielsweise

- bestimmte Eigenschaften ein- bzw. ausblenden. Neben den standardmäßig angezeigten Eigenschaften gibt es noch weitere Eigenschaften wie *Gruppen-ID* und Metadaten, die eingeblendet werden können. Besitzt ein Rule Service die zur Anzeige konfigurierten Metadaten nicht, dann werden die betreffenden Spalten in der Tabelle leer gelassen. (siehe Abschnitt 4.6.1, „Ein- und Ausblenden von Spalten“)
- zugehörige Spalten verschieben (siehe Abschnitt 4.6.2, „Verschieben einer Spalte“)
- das Sortierkriterium und die Sortierreihenfolge ändern (siehe Abschnitt 4.6.3, „Verändern des Sortierkriteriums und der Sortierreihenfolge“)

Weiterführende Konzepte.

- Abschnitt 3.1.1, „Rule Service“

Weiterführende Aufgaben.

- Abschnitt 4.2.2, „Filterung angezeigter Rule Services“
- Abschnitt 4.2.3, „Hinzufügen eines Rule Service mittels Visual Rules Archiv“

- Abschnitt 4.2.4, „Löschen eines bereitgestellten Rule Service“
- Abschnitt 4.2.5, „Anzeige der WSDL-Datei eines Rule Service“
- Abschnitt 4.6, „Konfiguration der Anzeige von Tabelleninhalten“

4.2.2. Filterung angezeigter Rule Services

Auf der Übersichtsseite der Sicht *Rule Services* können Sie Filterkriterien spezifizieren, um nur Rule Services, die von Interesse sind, anzuzeigen:

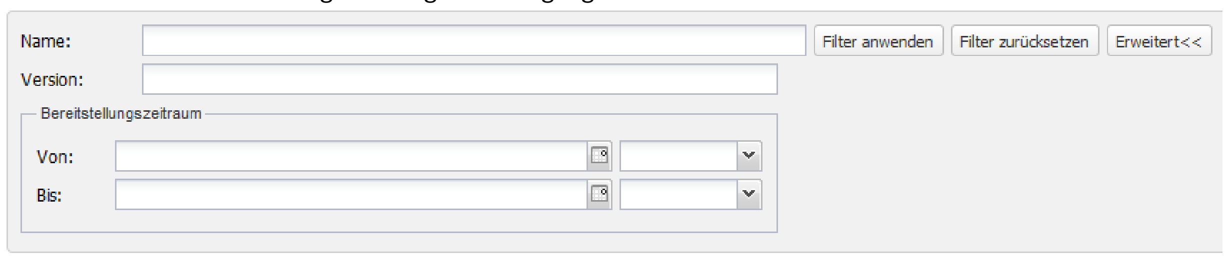


In diesem Abschnitt können Sie den Namen des Rule Service als Filterkriterium angeben.

Sie haben die Möglichkeit, weitere Filterkriterien für die Anzeige der Rule Services zu spezifizieren:

- Version
- Bereitstellungszeitraum

Dazu steht ein erweiterter Eingabedialog zur Verfügung:



Mehr Informationen zur Eingabe der Filterkriterien und der Anwendung des Filters finden Sie in Abschnitt 4.7, „Filterung angezeigter Objekte“.

4.2.3. Hinzufügen eines Rule Service mittels Visual Rules Archiv

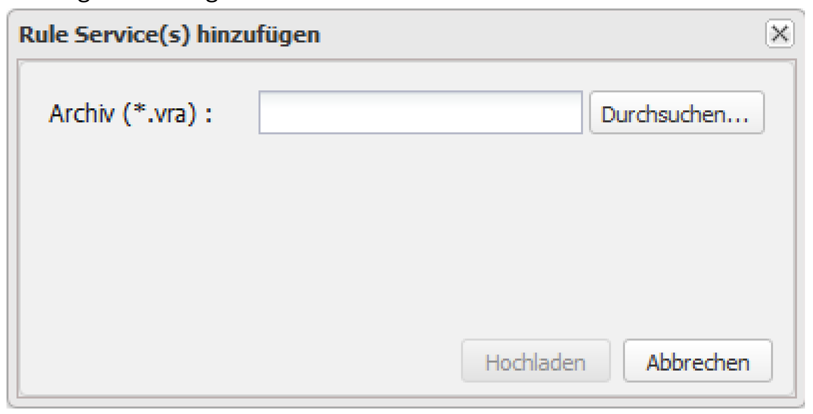
Der Execution Server unterstützt das Hochladen eines *Visual Rules Archivs*. Dieses enthält neben dem Rule Service selbst auch alle weiteren, zur Serviceausführung erforderlichen Bibliotheken.

1. Wählen Sie in der Webkonsole den Eintrag Rule Services aus, um die Sicht *Rule Services* zu öffnen.
2. Klicken Sie auf die Schaltfläche `Service(s) aus Archiv hinzufügen`.



Diese Schaltfläche ist nur dann aktiviert, wenn Sie ausreichende Berechtigungen haben. Der Standardbenutzer `Admin` hat diese Erlaubnis. Siehe auch Abschnitt 1.3, „Berechtigungs-Konzept“.

Der folgende Dialog erscheint:



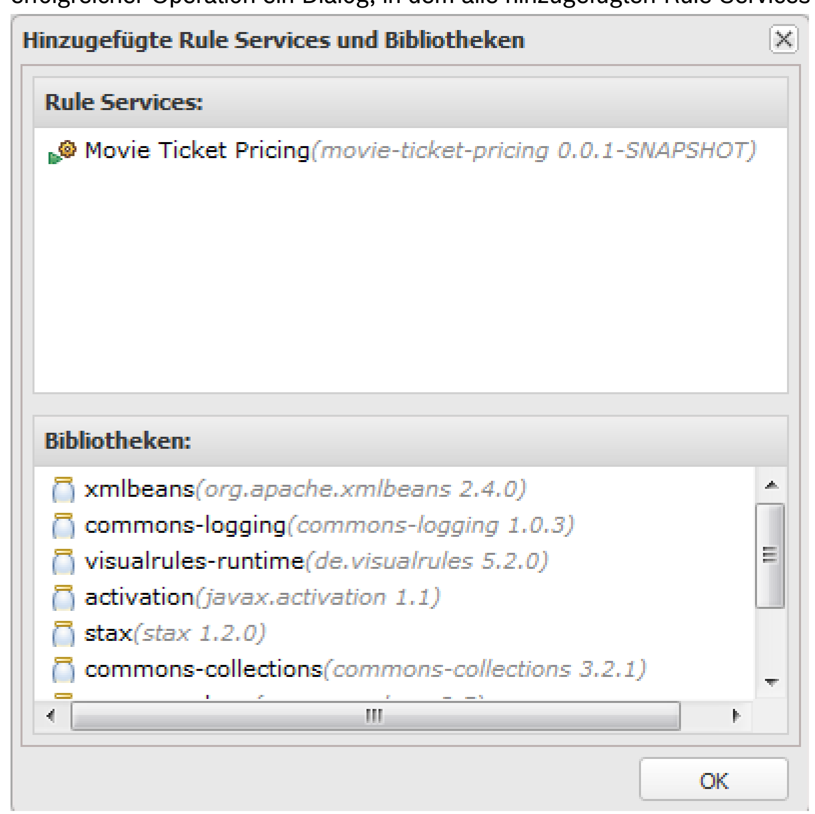
3. Klicken Sie auf die Schaltfläche Durchsuchen... .

   Es öffnet sich ein Dialog zum Hochladen des *Visual Rules Archivs*. Er ermöglicht Ihnen die Spezifikation einer *Visual Rules Archiv*-Datei (mit der Endung *.vra*). Das Aussehen des Dialoges ist abhängig von Ihrem Web Browser.

   Der Pfad der ausgewählten Datei wird anschließend im Eingabefeld angezeigt.

4. Klicken Sie auf die Schaltfläche Hochladen , um die spezifizierte Datei hochzuladen.

   Wenn die Operation abgeschlossen ist, wird eine Statusmeldung angezeigt. Zudem erscheint nach erfolgreicher Operation ein Dialog, in dem alle hinzugefügten Rule Services und Bibliotheken aufgelistet sind:



Weiterführende Konzepte.

- Abschnitt 3.1.4, „Visual Rules Archiv“

4.2.4. Löschen eines bereitgestellten Rule Service

1. Wählen Sie in der Webkonsole den Eintrag Rule Services aus, um die Sicht *Rule Services* zu öffnen.

2. Klicken Sie auf das Symbol des Rule Service, den Sie löschen möchten.



   Dieses Symbol ist nur verfügbar, wenn Sie ausreichende Berechtigungen haben. Der Standardbenutzer `Admin` hat diese Erlaubnis. Siehe auch Abschnitt 1.3, „Berechtigungs-Konzept“.

3. Es öffnet sich ein Dialog, in dem Sie bestätigen können, dass Sie die Aktion wirklich durchführen möchten.

   Klicken Sie zur Bestätigung auf die Schaltfläche Ja .

4.2.5. Anzeige der WSDL-Datei eines Rule Service

1. Wählen Sie in der Webkonsole den Eintrag Rule Services aus, um die Sicht *Rule Services* zu öffnen.
2. Klicken Sie auf das Symbol des Rule Service, dessen WSDL-Datei angezeigt werden soll.
3. Die WSDL-Datei wird im Web Browser geöffnet.

Falls Sie die Detailseite eines Rule Service geöffnet haben, können Sie auch dort die Anzeige der WSDL-Datei veranlassen:

1. Klicken Sie auf [WSDL anzeigen] im oberen Bereich der Detailseite.
2. Die WSDL-Datei wird im Web Browser geöffnet.

Die URL für die WSDL lautet:

```
http://<server>:<port>/<context-name>/services/<tenant-id>/<rule-model>/<version>/
<rule-model>.wsdl
```



Das WSDL importiert zusätzliche XML Schema-Dateien, welche wiederum andere Schema-Dateien importieren können. Alle diese Ressourcen können vom Execution Server heruntergeladen werden.

Sie können diese WSDL für jeden Web Service Client verwenden, um ein Regelmodell aufzurufen.

Weiterführende Konzepte.

- Abschnitt 5.1.6, „Abbildung des Regelmodells auf WSDL“

4.2.6. Herunterladen eines Rule Service

Sie können einen Rule Service als *Visual Rules Archiv* herunterladen. Ein *Visual Rules Archiv* enthält neben dem Rule Service auch noch alle weiteren, zur Serviceausführung erforderlichen Bibliotheken und kann auf einem anderen Visual Rules Execution Server hochgeladen werden.

1. Wählen Sie in der Webkonsole den Eintrag Rule Services aus, um die Sicht *Rule Services* zu öffnen.
2. Öffnen Sie die Detailseite des Rule Service, den Sie herunterladen wollen, indem Sie auf den zugehörigen Link klicken.
3. Klicken Sie im oberen Bereich der Detailseite auf [Visual Rules Archiv herunterladen].
4. Es öffnet sich ein Dialog zum Herunterladen der Datei. Er ermöglicht Ihnen das Speichern der Datei. Das Aussehen des Dialoges ist abhängig von Ihrem Web Browser.

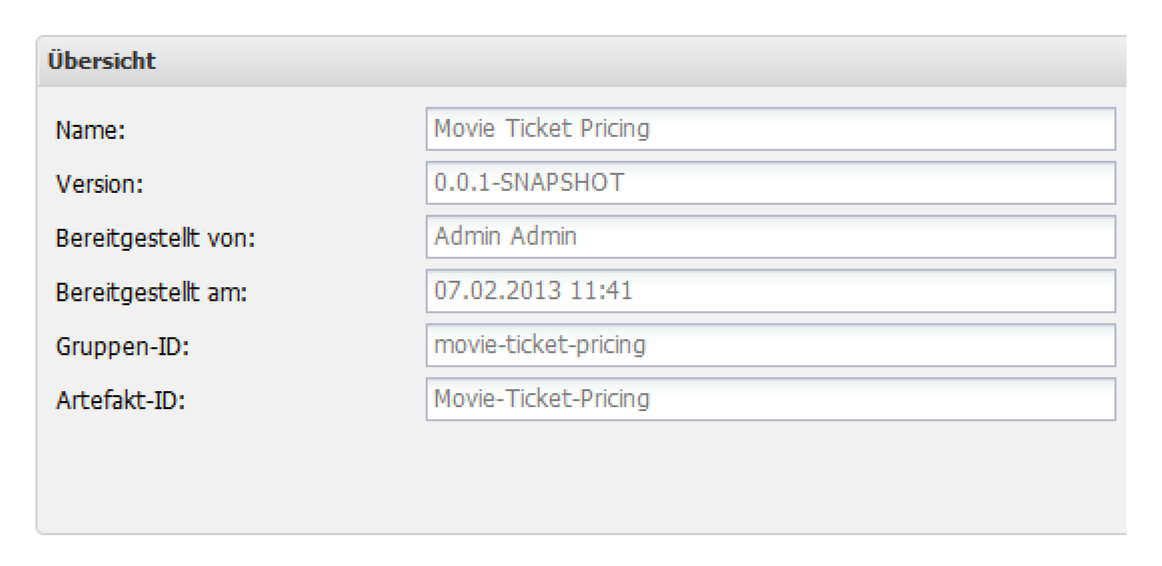
Weiterführende Konzepte.

- Abschnitt 3.1.2, „Regelbibliothek“
- Abschnitt 3.1.4, „Visual Rules Archiv“

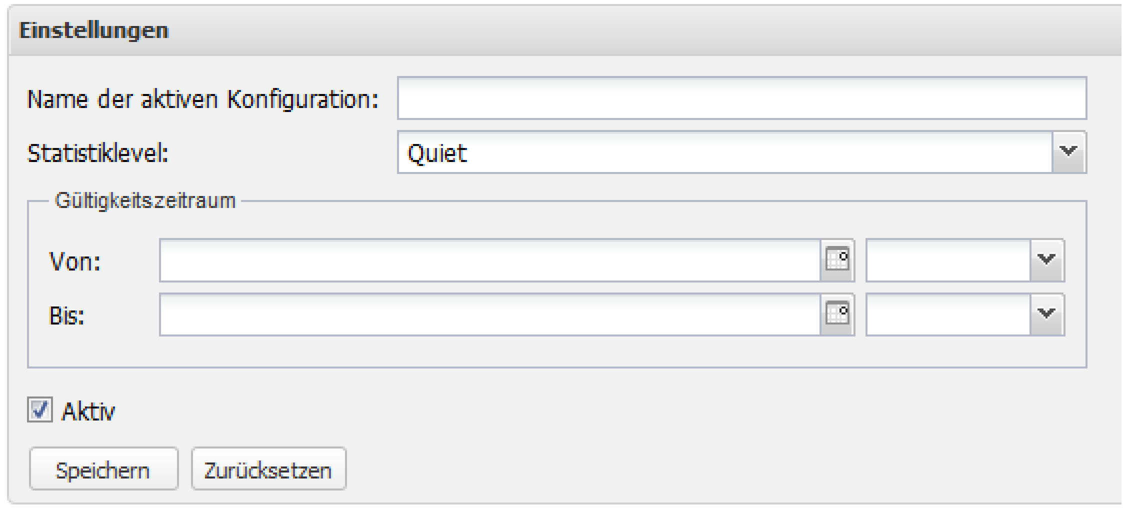
4.2.7. Anzeige der Eigenschaften und Änderung der Einstellungen eines Rule Service

1. Wählen Sie in der Webkonsole den Eintrag Rule Services aus, um die Sicht *Rule Services* zu öffnen.
2. Öffnen Sie die Detailseite des Rule Service, dessen Eigenschaften/Einstellungen Sie sehen wollen, indem Sie auf den zugehörigen Link klicken.

Auf der Detailseite werden im Abschnitt "**Übersicht**" allgemeine Eigenschaften des Rule Service angezeigt:



Neben den Eigenschaften werden auch (veränderbare) Einstellungen des Rule Service angezeigt (siehe Abschnitt 3.1.5, „Rule Service Einstellungen“):



Die Änderung einer Einstellung können Sie folgendermaßen vornehmen:

1. Führen Sie für die einzustellende Eigenschaft die entsprechende Eingabe/Auswahl aus:

   - *Name der aktiven Konfiguration* : Eingabe des Namens der zu verwendenden Konfiguration
   - *Statistiklevel* : Auswahl eines Statistiklevels aus der Dropdown-Liste
   - *Gültigkeitszeitraum* : Eingabe eines Start- und/oder Endzeitpunktes (siehe Abschnitt 4.7.1.1, „Eingabe von Datum und Uhrzeit“)
   - *Aktiv* : Selektieren/Deselektieren der CheckBox um den Rule Service zu aktivieren/deaktivieren

   Möchten Sie alle vorgenommenen Eingaben wieder auf die ursprünglich gespeicherten Werte zurücksetzen, dann klicken Sie auf die Schaltfläche Zurücksetzen .

2. Bestätigen Sie die Änderungen, indem Sie auf die Schaltfläche Speichern klicken.

Weiterführende Konzepte.

- Abschnitt 3.1.5, „Rule Service Einstellungen“

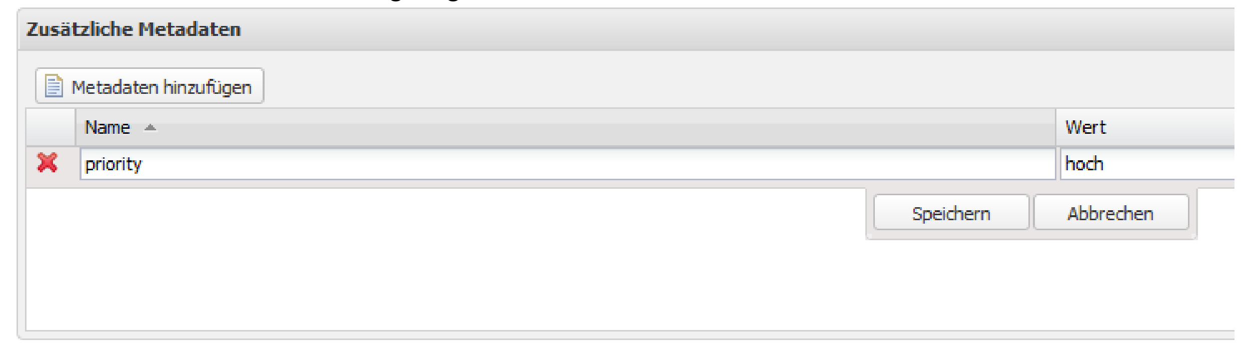
4.2.8. Verwaltung der Metadaten eines Rule Service

1. Wählen Sie in der Webkonsole den Eintrag Rule Services aus, um die Sicht *Rule Services* zu öffnen.
2. Öffnen Sie die Detailseite des Rule Service, dessen Metadaten Sie verwalten wollen, indem Sie auf den zugehörigen Link klicken.
3. Klicken Sie auf Metadaten, worauf die Metadatenseite geöffnet wird.

Es gibt *modellbezogene Metadaten* eines Rule Service, die im Regelmodell gespeichert sind und nach der Bereitstellung des Rule Service nicht mehr verändert/gelöscht werden können. Name und Wert dieser Metadaten werden auf der Metadatenseite des Rule Service im Abschnitt "**Modellbezogene Metadaten**" angezeigt:



Zudem gibt es *zusätzliche Metadaten*, die Sie über die Webkonsole einem Rule Service nach dessen Bereitstellung zuordnen können. Diese Metadaten sind editierbar und werden im Abschnitt "**Zusätzliche Metadaten**" der Metadatenseite angezeigt:



4.2.8.1. Hinzufügen von Metadaten

Sie haben die Möglichkeit, einem Rule Service weitere Metadaten hinzuzufügen:

1. Klicken Sie im Abschnitt "**Zusätzliche Metadaten**" auf die Schaltfläche Metadaten hinzufügen. Eine leere Tabellenzeile wird hinzugefügt.
2. Geben Sie in der Spalte **Name** den neuen Metadatennamen ein.
3. Geben Sie in der Spalte **Wert** den Metadatenwert ein.
4. Klicken Sie auf die Schaltfläche Speichern. Die Metadatenliste wird um diesen Eintrag erweitert.

4.2.8.2. Löschen von Metadaten

Löschbare Metadaten bekommen im Abschnitt "**Zusätzliche Metadaten**" das Symbol zugeordnet.

1. Klicken Sie auf das Symbol des Metadatums, das Sie löschen möchten.

2. Es öffnet sich ein Dialog, in dem Sie bestätigen können, dass Sie die Aktion wirklich durchführen möchten.

   Klicken Sie zur Bestätigung auf die Schaltfläche Ja.

4.2.8.3. Editieren von Metadatennamen und –werten

An den zusätzlichen Metadaten lassen sich der Name und der Wert nachträglich editieren.

1. Klicken Sie auf den Metadatennamen bzw. -wert, um die Eingabe direkt in der grafischen Darstellung vorzunehmen.
2. Geben Sie den Metadatennamen bzw. -wert ein.
3. Klicken Sie auf die Schaltfläche Speichern, um die Änderungen zu speichern.

Weiterführende Konzepte.

- Abschnitt 6.1, „Konzept Metadaten“

4.2.9. Anzeige der Ausführungen eines Rule Service

1. Wählen Sie in der Webkonsole den Eintrag Rule Services aus, um die Sicht *Rule Services* zu öffnen.
2. Öffnen Sie die Detailseite des Rule Service, dessen Ausführungen Sie sehen wollen, indem Sie auf den zugehörigen Link klicken.
3. Klicken Sie auf Ausführungen , worauf die Ausführungsseite geöffnet wird.

Die Ausführungsseite zeigt alle Aufrufe dieses Rule Service an. Zudem gibt es den Abschnitt "**Filter Ausführungen**", in dem Sie die Anzeige der Ausführungen einschränken können. Eine genauere Beschreibung finden Sie in Abschnitt 4.3.1, „Anzeige der Ausführungen von Rule Services“.

4.2.10. Anzeige der erforderlichen Bibliotheken eines Rule Service

1. Wählen Sie in der Webkonsole den Eintrag Rule Services aus, um die Sicht *Rule Services* zu öffnen.
2. Öffnen Sie die Detailseite des Rule Service, dessen erforderliche Bibliotheken Sie anzeigen möchten, indem Sie auf den zugehörigen Link klicken.
3. Klicken Sie auf Erforderliche Bibliotheken, worauf die Seite "*Erforderliche Bibliotheken*" geöffnet wird.

Die Seite "*Erforderliche Bibliotheken*" zeigt alle Bibliotheken an, die dieser Rule Service benötigt. Eine genauere Beschreibung finden Sie in Abschnitt 4.4.1, „Anzeige bereitgestellter Bibliotheken“.

4.3. Verwaltung der Ausführungen von Rule Services

In der Sicht *Ausführungen* des Visual Rules Execution Servers können Sie Ausführungen von Rule Services verwalten.

Wählen Sie in der Webkonsole den Eintrag Ausführungen, um diese Sicht zu öffnen.

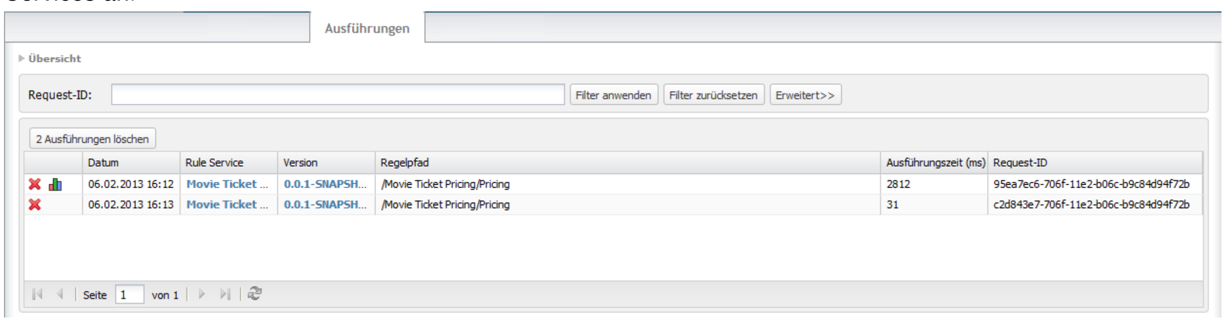
Hier haben Sie folgende Möglichkeiten:

- Abschnitt 4.3.1, „Anzeige der Ausführungen von Rule Services“
- Abschnitt 4.3.2, „Filterung angezeigter Ausführungen“
- Abschnitt 4.3.3, „Löschen von Ausführungen von Rule Services“
- Abschnitt 4.3.4, „Herunterladen einer Statistik zur Ausführung eines Rule Service“

4.3.1. Anzeige der Ausführungen von Rule Services

Wählen Sie in der Webkonsole den Eintrag Ausführungen aus, um die Sicht *Ausführungen* zu öffnen.

Die geöffnete Übersichtsseite zeigt die Aufrufe aller auf dem Visual Rules Execution Server bereitgestellten Rule Services an:



Standardmäßig werden folgende Eigenschaften der Ausführungen angezeigt:

- Ausführungszeitpunkt
- Name des aufgerufenen Rule Service
- Version des Rule Service
- Regelpfad
- Ausführungszeit
- Request-ID

Sie können das Layout der angezeigten Eigenschaften beeinflussen, indem Sie beispielsweise

- bestimmte Eigenschaften ein- bzw. ausblenden. Neben den standardmäßig angezeigten Eigenschaften gibt es noch weitere Eigenschaften wie *ID*, *Artefakt-ID* und *Gruppen-ID*, die eingeblendet werden können. (siehe Abschnitt 4.6.1, „Ein- und Ausblenden von Spalten“)
- zugehörige Spalten verschieben (siehe Abschnitt 4.6.2, „Verschieben einer Spalte“)
- das Sortierkriterium und die Sortierreihenfolge ändern (siehe Abschnitt 4.6.3, „Verändern des Sortierkriteriums und der Sortierreihenfolge“)

Falls bei der Ausführung eines Rule Service eine Statistik erstellt wurde, wird in der ersten Tabellenspalte das Symbol angezeigt.



Sind Sie nur an den Aufrufen/Ausführungen *eines* Rule Service interessiert, dann können Sie sich diese Ansicht über die Detailseite des Rule Service anzeigen lassen (siehe Abschnitt 4.2.9, „Anzeige der Ausführungen eines Rule Service“).

Weiterführende Aufgaben.

- Abschnitt 5.2.1, „Aufrufen eines Rule Service“
- Abschnitt 4.3.4, „Herunterladen einer Statistik zur Ausführung eines Rule Service“
- Abschnitt 4.3.3, „Löschen von Ausführungen von Rule Services“
- Abschnitt 4.6, „Konfiguration der Anzeige von Tabelleninhalten“

4.3.2. Filterung angezeigter Ausführungen

Auf der Übersichtsseite der Sicht *Ausführungen* können Sie Filterkriterien spezifizieren, um nur Ausführungen, die von Interesse sind, anzuzeigen:

Request-ID: [ ] Filter anwenden | Filter zurücksetzen | Erweitert>>

In diesem Abschnitt können Sie die Request-ID der Ausführung als Filterkriterium angeben.

Sie haben die Möglichkeit, weitere Filterkriterien für die Anzeige der Ausführungen zu spezifizieren:

- Name des aufgerufenen Rule Service (in der Ausführungsseite eines Rule Services nicht verfügbar)
- Version des aufgerufenen Rule Service (in der Ausführungsseite eines Rule Services nicht verfügbar)
- Regelpfad
- Ausführungszeitraum

Dazu steht ein erweiterter Eingabedialog zur Verfügung:

Request-ID: [ ] Filter anwenden | Filter zurücksetzen | Erweitert<<
Rule Service:
Version:
Regelpfad:
Ausführungszeitraum
Von:
Bis:

Mehr Informationen zur Eingabe der Filterkriterien und der Anwendung des Filters finden Sie in Abschnitt 4.7, „Filterung angezeigter Objekte“.

4.3.3. Löschen von Ausführungen von Rule Services

Sie haben die Möglichkeit, eine bestimmte Ausführung oder alle in der Sicht *Ausführungen* angezeigten Ausführungen zu löschen. Möchten Sie beispielsweise alle Ausführungen, die vor einem bestimmten Zeitpunkt liegen, löschen, dann können Sie sich durch Filterung der Ausführungen (Filterkriterium *Ausführungszeitraum*) die entsprechenden Ausführungen anzeigen lassen und darauf die Löschoperation ausführen.

1. Wählen Sie in der Webkonsole den Eintrag Ausführungen aus, um die Sicht *Ausführungen* zu öffnen.
2. Wenn Sie eine einzelne Ausführung löschen möchten, dann klicken Sie auf das Symbol ✖ der Ausführung, die Sie löschen möchten

   oder

   wenn Sie alle aktuell angezeigten Ausführungen löschen möchten, dann klicken Sie dort auf die Schaltfläche **Löschen**.
3. Es öffnet sich ein Dialog, in dem Sie bestätigen können, dass Sie die Aktion wirklich durchführen möchten.

   Klicken Sie zur Bestätigung auf die Schaltfläche Ja.

Weiterführende Aufgaben.

- Abschnitt 4.3.1, „Anzeige der Ausführungen von Rule Services“

4.3.4. Herunterladen einer Statistik zur Ausführung eines Rule Service

1. Wählen Sie in der Menüleiste der Webkonsole den Eintrag Ausführungen aus, um die Sicht *Ausführungen* zu öffnen.

2. Klicken Sie bei der Ausführung, deren Statistik Sie herunterladen wollen, auf das Symbol . Das Symbol wird in der ersten Tabellenspalte nur angezeigt, wenn bei der Ausführung eine Statistik erstellt wurde.

3. Es öffnet sich ein Dialog zum Herunterladen der *vrstatistic*-Datei. Er ermöglicht Ihnen das Speichern der Datei. Das Aussehen des Dialoges ist abhängig von Ihrem Web Browser.

   Die heruntergeladene Statistik kann vom Visual Rules Modeler geladen und angezeigt werden.

Weiterführende Aufgaben.

- Abschnitt 4.3.1, „Anzeige der Ausführungen von Rule Services“

4.4. Verwaltung bereitgestellter Bibliotheken

In der Sicht *Bibliotheken* des Visual Rules Execution Servers können Sie bereitgestellte Bibliotheken verwalten.

Wählen Sie in der Webkonsole den Eintrag Bibliotheken, um diese Sicht zu öffnen.

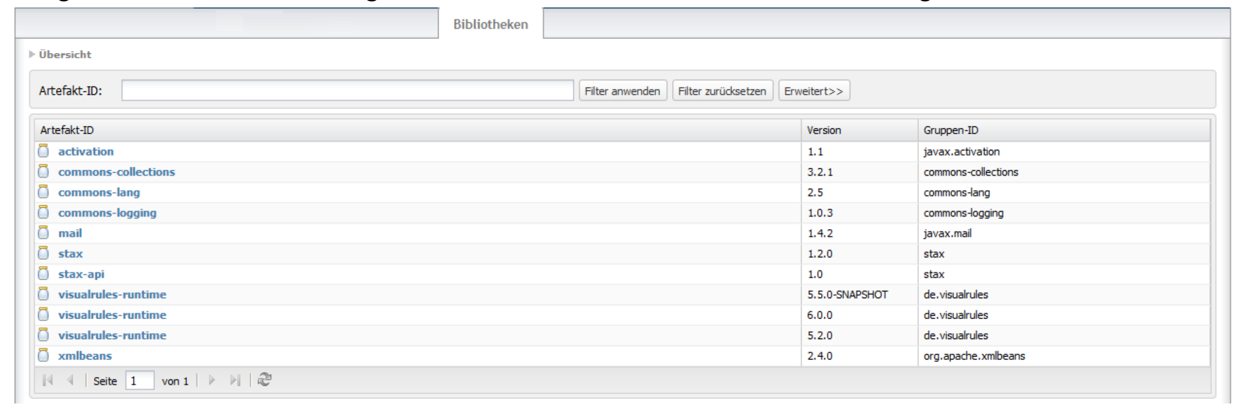
Hier haben Sie folgende Möglichkeiten:

- Abschnitt 4.4.1, „Anzeige bereitgestellter Bibliotheken“
- Abschnitt 4.4.2, „Filterung angezeigter Bibliotheken“
- Abschnitt 4.4.3, „Anzeige von Eigenschaften und Verwendung einer Bibliothek“
- Abschnitt 4.4.4, „Löschen einer bereitgestellten Bibliothek“

4.4.1. Anzeige bereitgestellter Bibliotheken

Wählen Sie in der Webkonsole den Eintrag Bibliotheken aus, um die Sicht *Bibliotheken* zu öffnen.

Die geöffnete Übersichtsseite zeigt alle auf dem Visual Rules Execution Server bereitgestellten Bibliotheken an:



Standardmäßig werden folgende Eigenschaften der Bibliotheken angezeigt:

- Artefakt-ID
- Version
- Gruppen-ID

Sie können das Layout der angezeigten Eigenschaften beeinflussen, indem Sie beispielsweise

- bestimmte Eigenschaften ein- bzw. ausblenden. Neben den standardmäßig angezeigten Eigenschaften gibt es noch weitere Eigenschaften wie *Bereitgestellt von* und *Bereitgestellt am*, die eingeblendet werden können. (siehe Abschnitt 4.6.1, „Ein- und Ausblenden von Spalten“)
- zugehörige Spalten verschieben (siehe Abschnitt 4.6.2, „Verschieben einer Spalte“)
- das Sortierkriterium und die Sortierreihenfolge ändern (siehe Abschnitt 4.6.3, „Verändern des Sortierkriteriums und der Sortierreihenfolge“)



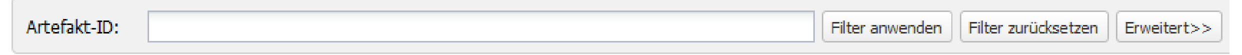
Sind Sie nur an den Bibliotheken interessiert, die *ein bestimmter* Rule Service benötigt, dann können Sie sich diese Ansicht über die Detailseite des Rule Service anzeigen lassen (siehe Abschnitt 4.2.10, „Anzeige der erforderlichen Bibliotheken eines Rule Service“).

Weiterführende Aufgaben.

- Abschnitt 4.4.4, „Löschen einer bereitgestellten Bibliothek“
- Abschnitt 4.6, „Konfiguration der Anzeige von Tabelleninhalten“

4.4.2. Filterung angezeigter Bibliotheken

Auf der Übersichtsseite der Sicht *Bibliotheken* können Sie Filterkriterien spezifizieren, um nur Bibliotheken, die von Interesse sind, anzuzeigen:

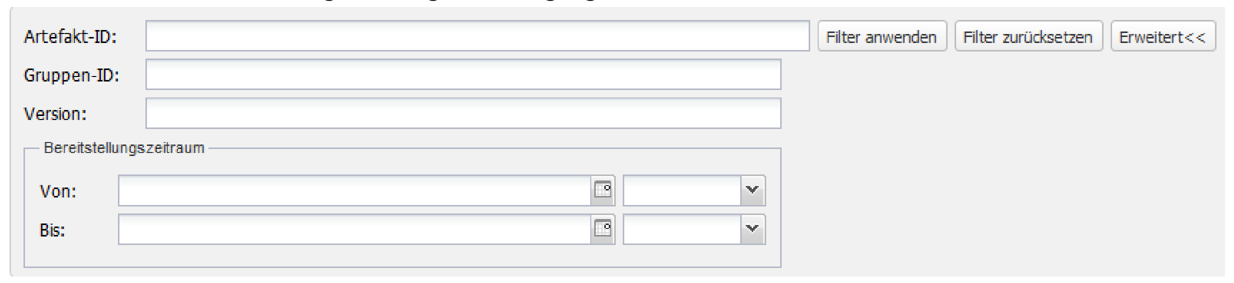


In diesem Abschnitt können Sie die Artefakt-ID der Bibliothek als Filterkriterium angeben.

Sie haben die Möglichkeit, weitere Filterkriterien für die Anzeige der Bibliotheken zu spezifizieren:

- Gruppen-ID
- Version
- Bereitstellungszeitraum

Dazu steht ein erweiterter Eingabedialog zur Verfügung:

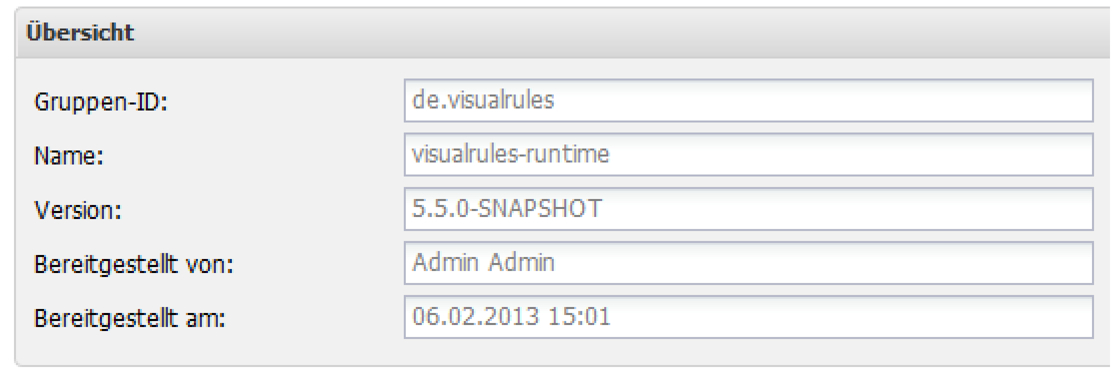


Mehr Informationen zur Eingabe der Filterkriterien und der Anwendung des Filters finden Sie in Abschnitt 4.7, „Filterung angezeigter Objekte“.

4.4.3. Anzeige von Eigenschaften und Verwendung einer Bibliothek

1. Wählen Sie in der Webkonsole den Eintrag Bibliotheken aus, um die Sicht *Bibliotheken* zu öffnen.
2. Öffnen Sie die Detailseite der Bibliothek, deren Eigenschaften/Verwendung Sie sehen möchten, indem Sie auf den zugehörigen Link klicken.

Auf der Detailseite werden im Abschnitt "**Übersicht**" allgemeine Eigenschaften der Bibliothek angezeigt:



Neben den Eigenschaften werden auf dieser Seite im Abschnitt "**Rule Services**" alle Rule Services angezeigt, die diese Bibliothek verwenden:


Rule Services

Rule Service	Artefakt-ID	Version	Bereitgestellt von	Bereitgestellt am
Coffee vending machine	Coffee-vending...	0.0.1-SNAPSHOT	Admin Admin	06.02.2013 15:01

Seite 1 von 1

Eine genauere Beschreibung der Tabelle finden Sie in Abschnitt 4.2.1, „Anzeige bereitgestellter Rule Services“.

4.4.4. Löschen einer bereitgestellten Bibliothek

1. Wählen Sie in der Webkonsole den Eintrag Bibliotheken aus, um die Sicht *Bibliotheken* zu öffnen.
2. Öffnen Sie die Detailseite der Bibliothek, die Sie löschen wollen, indem Sie auf den zugehörigen Link klicken.
3. Klicken Sie auf die Schaltfläche **Ungenutzte Bibliothek löschen**.



Diese Schaltfläche ist nur dann aktiviert, wenn diese Bibliothek von keinem anderen Rule Service mehr benötigt wird und wenn Sie ausreichende Berechtigungen haben. Der Standardbenutzer `Admin` hat diese Erlaubnis. Siehe auch Abschnitt 1.3, „Berechtigungs-Konzept“.

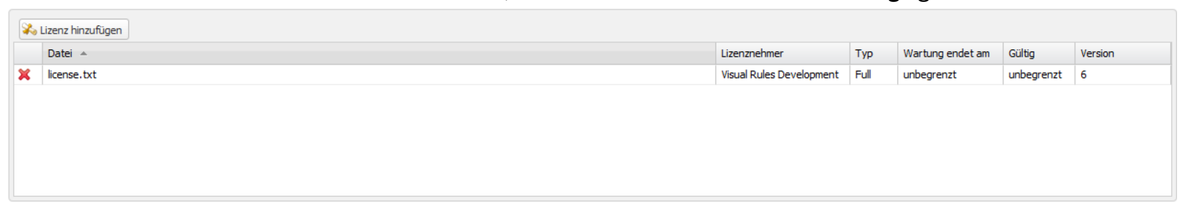
4. Es öffnet sich ein Dialog, in dem Sie bestätigen können, dass Sie die Aktion wirklich durchführen möchten.

   Klicken Sie zur Bestätigung auf die Schaltfläche Ja.

4.5. Wartung des Execution Servers

4.5.1. Lizenzen verwalten

1. Wählen Sie in der Webkonsole den Eintrag Wartung aus, um die Sicht *Wartung* zu öffnen.
2. Klicken Sie auf die Schaltfläche Lizenz-Verwaltung , worauf die Seite "*Lizenz-Verwaltung*" geöffnet wird:


Lizenz hinzufügen

Datei	Lizenznehmer	Typ	Wartung endet am	Gültig	Version
license.txt	Visual Rules Development	Full	unbegrenzt	unbegrenzt	6

Sie zeigt die Lizenzdateien für den Visual Rules Execution Server. Dabei sind für jede Lizenzdatei folgende Informationen angegeben:

- Name der Lizenzdatei
- Lizenznehmer
- Lizenztyp
- Wartungsende
- Gültigkeit
- Version

Außerdem haben Sie die folgenden Möglichkeiten:

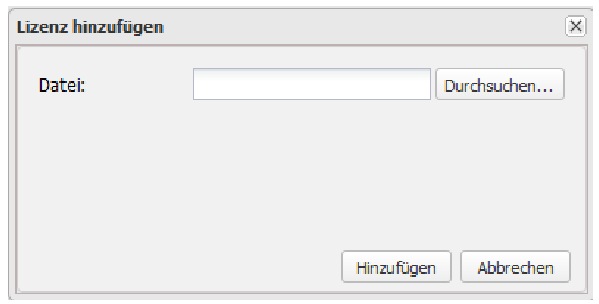
- Abschnitt 4.5.1.1, „Lizenz hinzufügen“

- Abschnitt 4.5.1.2, „Lizenz löschen“

4.5.1.1. Lizenz hinzufügen

Um eine Lizenz hinzuzufügen, gehen Sie vor wie folgt:

1. Klicken Sie auf der Seite "*Lizenz-Verwaltung*" auf **Lizenz hinzufügen**.
2. Der folgende Dialog erscheint:



3. Klicken Sie auf **Durchsuchen** und wählen Sie eine Lizenzdatei von Ihrem Dateisystem.
4. Klicken Sie auf **Hinzufügen**.

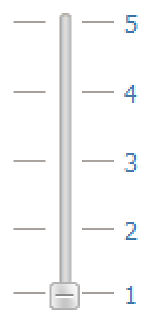
4.5.1.2. Lizenz löschen

Um eine Lizenz zu löschen, klicken Sie auf der Seite "*Lizenz-Verwaltung*" auf vor der betreffenden Lizenz.

4.5.2. Konfiguration und Anzeige von Nachrichten der Laufzeitprotokollierung

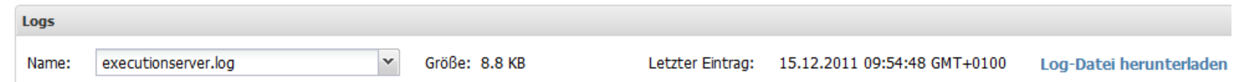
Wählen Sie in der Menüleiste der Webkonsole den Eintrag Wartung aus, um die Sicht *Wartung* zu öffnen. Klicken Sie danach auf **Logging**.

Die geöffnete Seite besteht aus zwei Teilen. Der obere, mit "**Detaillierungsgrad**" beschriebene Teil, dient zur Konfiguration der Laufzeitprotokollierung des Execution Server. Dazu dient ein Schieberegler:



Der Schieberegler kontrolliert die Einstellung des Detaillierungsgrades der Laufzeitprotokollierung. Je höher der Grad, desto mehr Nachrichten werden produziert. Zu beachten ist, dass ein hoher Detaillierungsgrad einen negativen Einfluss auf die Leistungsfähigkeit des Systems hat. Neben dem Schieberegler wird eine genaue Beschreibung für den momentan eingestellten Detaillierungsgrad angezeigt.

Der untere Teil der Seite ist mit "**Logs**" beschrieben und zeigt hauptsächlich Protokolldateien (Log) an:



Die Dropdown-Liste wählt die Protokolldatei aus, deren Einträge angezeigt werden sollen. Eine Datei kann auch leer sein. Informationen zu existierenden Protokolldateien ist in Abschnitt 2.7.3, „Laufzeitprotokollierung“

beschrieben. Mit dem "**Log-Datei herunterladen**" beschrifteten Link, kann die ausgewählte Protokolldatei heruntergeladen werden.

4.6. Konfiguration der Anzeige von Tabelleninhalten

Die tabellarische Anzeige von Informationen in der Webkonsole kann oftmals auf folgende Art und Weise verändert werden:

- Abschnitt 4.6.1, „Ein- und Ausblenden von Spalten“
- Abschnitt 4.6.2, „Verschieben einer Spalte“
- Abschnitt 4.6.3, „Verändern des Sortierkriteriums und der Sortierreihenfolge“

4.6.1. Ein- und Ausblenden von Spalten

In vielen Tabellen ist es möglich, die Anzeige bestimmter Spalten zu aktivieren/deaktivieren:

1. Bewegen Sie die Maus über die Überschrift einer beliebigen Spalte der Tabelle. Klicken Sie auf , um das Kontextmenü zu öffnen.
2. Gehen Sie mit der Maus auf den Menüpunkt Spalten, worauf sich ein weiteres Kontextmenü öffnet, in dem ausblendbare Spalten aufgeführt werden.
3. Aktivieren/Deaktivieren Sie die Checkbox der Spalten, die Sie ein-/ausblenden möchten.

4.6.2. Verschieben einer Spalte

In vielen Tabellen haben Sie die Möglichkeit, die Reihenfolge der Spalten zu verändern:

1. Eine Spalte kann per Drag&Drop an eine andere Position verschoben werden. Klicken Sie auf die Überschrift der Spalte, die Sie verschieben möchten, und halten Sie die linke Maustaste gedrückt.
2. Bewegen Sie die Maus nach links bzw. rechts. Während der Mausbewegung erscheinen Symbole, die signalisieren, ob es sich bei der aktuellen Position um eine mögliche oder unerlaubte Position handelt. Zusätzlich zeigen Pfeile die genaue (mögliche) Position an.
3. Haben Sie die gewünschte Spaltenposition erreicht, dann lassen Sie die Maustaste los. Die Spalte wird nun dorthin verschoben.

4.6.3. Verändern des Sortierkriteriums und der Sortierreihenfolge

1. Klicken Sie auf die Spaltenüberschrift, um die zugehörige Eigenschaft als Sortierkriterium festzulegen. Das Symbol bedeutet, dass *aufsteigend* sortiert wird.
2. Falls Sie eine absteigende Sortierreihenfolge möchten, dann klicken Sie erneut auf die Spaltenüberschrift. Das Symbol bedeutet, dass *absteigend* sortiert wird.



Alternativ können Sie die Sortierreihenfolge auch über das Kontextmenü der Spalte festlegen:

1. Bewegen Sie die Maus über die Spaltenüberschrift und klicken Sie auf , um das Kontextmenü zu öffnen.
2. Selektieren Sie den Menüpunkt Aufsteigend sortieren bzw. Absteigend sortieren.

4.7. Filterung angezeigter Objekte

Durch die Anwendung eines Filters kann die Zahl der angezeigten Objekte in einer Tabelle reduziert und der Fokus auf bestimmte Objekte gelegt werden.

4.7.1. Eingabe von Filterkriterien

Filterkriterien werden in einem eigenen Abschnitt der Benutzeroberfläche spezifiziert. Der Filterabschnitt befindet sich meist vor der Tabelle, auf deren Objekte sich die Filterung beziehen soll. Initial ist die Eingabe *eines* (wichtigen) Filterkriteriums möglich.

Um weitere Filterkriterien eingeben zu können, können Sie in einen erweiterten Eingabedialog mit zusätzlichen Eingabefeldern wechseln. Dazu gibt es im Filterabschnitt die Schaltfläche Erweitert>>.

Den erweiterten Dialog können Sie wieder verlassen, indem Sie dort auf die Schaltfläche Erweitert<< klicken.

Sie können die Eingabe aller Filterkriterien rückgängig machen, indem Sie auf die Schaltfläche Filter zurücksetzen klicken. Die Filtereingabefelder sind danach leer. Zudem wird die Anzeige der zugehörigen Tabelle aktualisiert.

4.7.1.1. Eingabe von Datum und Uhrzeit

Die Eingabefelder zur Spezifikation eines Datums und einer Uhrzeit haben folgendes Aussehen:

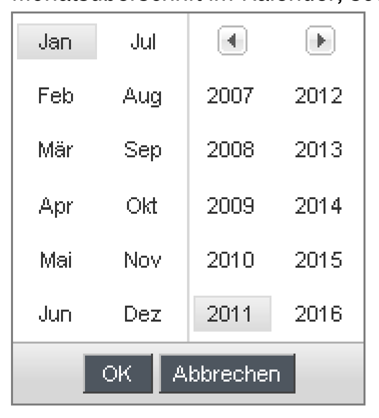


Zur Eingabe eines Datums klicken Sie auf das Symbol , sodass sich ein Kalender öffnet:



Hier können Sie per Mausklick ein Datum auswählen, das dann automatisch in das Datumsfeld übernommen wird.

Der Kalender bietet auch die Möglichkeit, den Monat und das Jahr zu wechseln. Klicken Sie dazu auf die Monatsüberschrift im Kalender, sodass sich folgende Ansicht zur Auswahl öffnet:



In dieser Ansicht können Sie mit der Maus einen Monat in einem bestimmten Jahr auswählen. Bestätigen Sie Ihre Wahl anschließend mit **OK**.

Zur Eingabe einer Uhrzeit klicken Sie auf das Symbol neben dem zweiten Eingabefeld. Es öffnet sich eine Liste, aus der Sie einen Zeitpunkt (Stunden und Minuten) auswählen können.

4.7.2. Anwenden eines Filters

Soll die Filterung der Objekte einer Tabelle ausgeführt werden, klicken Sie im Filterabschnitt auf die Schaltfläche Filter anwenden. Anschließend wird die Anzeige der zugehörigen Tabelle aktualisiert und es werden nur solche Objekte angezeigt, deren Eigenschaften mit den spezifizierten Filterkriterien übereinstimmen.

Kapitel 5. Aufrufen von Regeln im Execution Server

5.1. Konzepte

SOAP Nachrichten werden verwendet, um Rule Services im Execution Server aufzurufen. Der `SOAP Envelope` einer Nachricht umfasst einen `SOAP Header`, der Authentifizierungsdaten beinhaltet, und einen `SOAP Body`, der zur Spezifikation der Rule Service Anfrage verwendet wird.

5.1.1. Authentifizierung für eine Rule Service Anfrage

Der `SOAP Header` enthält Elemente, die in der Spezifikation der *Web Service Security* definiert sind. Es gibt ein `Security` Element zur Übermittlung sicherheitsrelevanter Informationen in einer *SOAP Nachricht*. Das enthaltene `UsernameToken` Element dient dazu, einen Benutzernamen und zusätzliche Attribute, die zur Authentifizierung nötig sind (Passwort und Mandant des Benutzers), zur Verfügung zu stellen.

Beispielsweise kann dies so aussehen:

```
<soapenv:Envelope ...>
   <soapenv:Header xmlns:wsse=
           "http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <wsse:Security soapenv:mustUnderstand="1">
         <wsse:UsernameToken>
            <tenant>BOSCH</tenant>
            <wsse:Username>USER</wsse:Username>
            <wsse:Password Type=
               "http://www.visual-rules.com/wss#PasswordText">PASSWORD</wsse:Password>
         </wsse:UsernameToken>
      </wsse:Security>
   </soapenv:Header>
   <soapenv:Body>
    ...
   </soapenv:Body>
</soapenv:Envelope>
```

Beispiel 5.1. SOAP Envelope mit Header zur Authentifizierung

Weiterführende Konzepte.

- Abschnitt 1.2, „Identity Management“

5.1.2. Format der Rule Service Anfrage

Das äußerste Element einer Anfrage an eine Regel wird `VRRequest` genannt. Der Namespace dieses Elements reflektiert die Position der Regel im Regelmodell und in der Regelpakethierarchie. Diese Information wird vom Execution Server verwendet, um die Anfrage an das gewünschte Regelmodell und an die darin gewünschte Regel weiterzuleiten.



Der vollständige Namespace für den URI des `VRRequest` besteht aus dem Präfix, der im Regelmodell konfiguriert werden kann, gefolgt vom Segment `/vrpath`, gefolgt von dem Pfad der Regel, die aufgerufen werden soll. Der Pfad besteht aus dem Namen des Regelmodells, desweiteren den Namen der Regelpakete und schliesslich des Namens der Regel selbst.

Der Namespace URI Präfix ist standardmäßig `http://www.visual-rules.com` und wird in den folgenden Beispielen verwendet.



Die nachfolgende Beschreibung bezieht sich auf die aktuelle Version. Bei der Verwendung von migrierten Rule Services ist die Web Service Schnittstelle immer noch richtig, enthält aber nicht alle hier beschriebenen Elemente.

Das `VRRequest` Element kann ein optionales `target` Element beinhalten, mit dem eine spezifische Version des aufzurufenden Regelmodells angeben werden kann. Wenn das `target` Element nicht vorhanden ist, ist es das Standardverhalten des Execution Server, die neueste Version des Regelmodells aufzurufen.

Als nächstes kommt das `configuration` Element, dass Einstellungen zur Ausführung des Rule Services spezifiziert und ebenfalls optional ist.



Alle Elemente haben eine definierte Reihenfolge. Ist das `target` Element spezifiziert, so muss es das erste Element im `VRRequest` sein und das `configuration` Element käme als nächstes, wenn es spezifiziert ist.

Schließlich werden die Eingabedaten der Regel in einem Element `input` spezifiziert. Die Elemente in diesem Abschnitt haben den gleichen Namen wie die Eingaben einer Regel. Der Wert jedes Datenelements wird innerhalb dieser Elemente spezifiziert. Der Abschnitt 5.1.5, „XML Repräsentation von Datentypen" beschreibt die XML Repräsentation für diese Werte, abhängig von den Datentypen (einschließlich Strukturen, Aufzählungen, Listen, Mengen und Maps).

```
<pricing:VRRequest xmlns:vr="http://www.visual-rules.com"
           xmlns:pricing="http://www.visual-rules.com/vrpath/Movie%20Ticket%20Pricing/Pricing">
   <!-- Falls weggelassen, wird Visual Rules Execution Server die
        "letzte" Version verwenden -->
   <vr:target>
      <version>1.0.1</version>
   </vr:target>
   <!-- Spezifiziert optionale Einstellungen für die Ausführung eines Rule Service. -->
   <vr:configuration>
      <!-- Eine requestId kann spezifiziert werden, um die Statistik auf dem Server
           zu identifizieren. Falls weggelassen, wird eine eindeutige ID generiert. -->
      <requestId>uniqueRequestId</requestId>

      <!-- Der Name der aktiven Konfiguration (active configuration name) der während der
           Ausführung verwendet werden soll.
           Wird dieser weggelassen, wird der Standard verwendet. -->
      <activeConfigurationName>test</activeConfigurationName>

      <!-- Für ältere Aufrufer, kann anstatt activeConfigurationName auch binding noch
           verwendet werden. -->
      <!-- Ebenso sollte dies für Rule Services verwendet werden, die mit einer
           Version vor 5.2 gebaut wurde. -->
      <binding>test</binding>

      <!-- Der verwendete Statistik Level. Mögliche Angaben sind: high, medium und low -->
      <!-- Falls weggelassen, werden keine Statistiken aufgezeichnet und sind
           auch nicht herunterladbar. -->
      <sessionStatistics>
          <level>medium</level>
      </sessionStatistics>
   </vr:configuration>
   <input>
      <auditorium_no>1</auditorium_no>
      <seat_no>199</seat_no>
      <show_date>2008-08-22</show_date>
      <coupon>true</coupon>
      <student>true</student>
      <bonus_card>GOLD</bonus_card>
   </input>
</pricing:VRRequest>
```

Beispiel 5.2. VRRequest Format

Weiterführende Konzepte.

- Abschnitt 5.1.5, „XML Repräsentation von Datentypen"

5.1.3. Format der Rule Service Antwort

Das äußerste Element einer Antwort auf eine Regel Anfrage wird `VRResponse` genannt. Der Namespace ist der gleiche wie für die Anfrage.

Die `VRResponse` hat ein Element mit dem Namen `output`, dass die Werte für alle Ausgabedatenelemente einer Regel enthält. Im Normalfall enthält die Antwort auch Aktionen. Es ist zu beachten, dass dies konfiguriert werden

kann, wie in Abschnitt 3.2.3, „Einstellen von Aktionen als Rückgabewert“ erläutert. Der Abschnitt 5.1.5, „XML Repräsentation von Datentypen“ beschreibt die XML Repräsentation für die Werte hier.

Schließlich gibt es ein Element `trace`, das die Informationen darüber zurückgibt, welche Regel in welcher Version von welchem Regelmodell gerade aufgerufen wurde. Dies dient dem Zweck der Nachverfolgbarkeit, so dass immer klar erkenntlich ist, welche Regeln ausgeführt wurden, beispielsweise wenn in der Anfrage die zu verwendende Version nicht festgelegt wurde.

Um die Statistiken nach der Ausführung auf dem Execution Server zu identifizieren, wird die angegebene oder generierte `requestId` zurückgeliefert. Mehr zu Statistiken ist in Abschnitt 4.3.1, „Anzeige der Ausführungen von Rule Services“ zu finden. Als letztes Element wird der Name der aktiven Konfiguration (`active configuration name`) ausgegeben, der entweder in der Anfrage oder als Einstellung des Rule Service spezifiziert wurde. Auch hier wird aus Kompatibilitätsgrüunden für ältere Rule Services das Element `binding` zurückgeliefert.

```
<pricing:VRResponse xmlns:vr="http://www.visual-rules.com"
           xmlns:pricing="http://www.visual-rules.com/vrpath/Movie%20Ticket%20Pricing/Pricing">
   <output>
      <price>7</price>
   </output>
   <vr:trace>
      <ruleModel>Movie Ticket Pricing</ruleModel>
      <rulePath>/Movie Ticket Pricing/Pricing</rulePath>
      <version>1.0.1</version>
      <requestId>3919c5b8-5bfc-11de-b728-d56c9d794642</requestId>
      <!-- Optional -->
      <activeConfigurationName>test</activeConfigurationName>
      <!-- oder -->
      <binding>test</binding>
   </vr:trace>
</pricing:VRResponse>
```

Beispiel 5.3. VRResponse Format

Weiterführende Konzepte.

- Abschnitt 5.1.5, „XML Repräsentation von Datentypen“

5.1.4. Generische Rule Service Anfragen

Der gewöhnliche Weg um einen Rule Service aufzurufen, besteht darin eine Anfrage aus der spezifischen WSDL zu erzeugen und diese abzuschicken. Dieses Vorgehen bedeutet auch, dass sich der Aufrufer bereits für einen Rule Service und den damit verbundenen Regeln entschieden hat. Es gibt jedoch auch Fälle, in denen es gewünscht ist, gegen eine für alle Rule Services gleiche WSDL zu arbeiten und die Anfrage daraus zu erstellen. Dies setzt eine implizite Kenntniss der Daten, die zu übergeben sind, voraus. Die zweite Anwendung der generischen Anfrage ist das Weiterleiten einer Anfrage zu einem Rule Service anhand von Metadaten.



Es ist ebenso möglich eine spezifische WSDL anhand von Metadaten abzuholen. Mehr dazu in Abschnitt 6.2.1.2, „Eine WSDL durch Angabe von Metadata abholen“

5.1.4.1. Generisches VRRequest Format

Der generische VRRequest ist in einer speziellen WSDL Datei spezifiziert, die auf dem Execution Server unter `http://<server>:<port>/<context-name>/services/visualrules_generic.wsdl` verfügbar ist. In einer Standard Installation auf einem lokalen Rechner könnte dies beispielsweise so aussehen (wobei x und y der Versionsnummer der WAR Datei des Execution Servers entsprechen):

```
http://localhost:8080/executionserver-6.x.y/services/visualrules_generic.wsdl
```

Es gibt einige Ähnlichkeiten zwischen dem spezifischen und dem generischen Anfrage Format. Das `VRRequest` Element ist ebenfalls das äußerste Element in der generischen Anfrage allerdings mit einem festen Namespace. Weil die Information des Regel Modells und der aufzurufenden Regel nicht mehr im Namespace vorhanden sind, müssen diese nun mit `ruleModel` und `rulePath` im `target` Element spezifiziert werden. Deswegen ist `target` auch nicht mehr optional. Es kann optional eine `version` angegeben werden, was die gleiche Bedeutung wie im spezifischen `VRRequest` hat.

Als Alternative zur `version` kann ein Element `effectiveDate` angegeben werden, welches die Anfrage anhand der Metadaten zu einem Rule Service leitet. In Abschnitt 6.2.1.1, „Standard-Metadata Mapper“ ist die Funktionsweise genauer erklärt. Zu beachten ist, dass die Elemente, welche unter `target` eingetragen werden ebenfalls angepasst werden können, wenn ein benutzerdefinierter `Matadata Mapper` eingesetzt wird, wie das in Abschnitt 6.2.1.3, „Benutzerdefinierter Metadata Mapper“ beschrieben ist.

Das nächste Element ist `configuration`, dass genau dasselbe ist, wie in Abschnitt 5.1.2, „Format der Rule Service Anfrage“ beschrieben. Als nächstes kommt das `input` Element, welches beliebige Elemente akzeptiert. Das ist der Nachteil des generischen Ansatzes und dies ist zugleich der größte Unterschied zur spezifischen Anfrage, die genau beschriebt, welche Elemente und Typen enthalten sind.

```
<gen:VRRequest xmlns:vr="http://www.visual-rules.com"
               xmlns:gen="http://www.visual-rules.com/generic" >
   <target>
      <ruleModel>Movie Ticket Pricing</ruleModel>
      <rulePath>/Movie Ticket Pricing/Pricing</rulePath>
      <!--Eines der beiden Elemente kann angegeben werden:-->
      <effectiveDate>2009-10-03</effectiveDate>
      <!-- oder -->
      <version>1.0.1</version>
   </target>
   <!-- Optional -->
   <vr:configuration>
      <!-- Die gleichen Elemente wie in der spezifischen Anfrage -->
   </vr:configuration>
   <input>
      <!-- Diese Elemente werden nicht vom XML Schema definiert. Die hier verwendeten
           Daten und Typen müssen vom Regelinterface akzeptiert werden. -->
      <auditorium_no>1</auditorium_no>
      <seat_no>199</seat_no>
      <show_date>2008-08-22</show_date>
      <coupon>true</coupon>
      <student>true</student>
      <bonus_card>GOLD</bonus_card>
   </input>
</gen:VRRequest>
```

Die Antwort einer generischen Anfrage unterscheidet sich nicht von der einer spezifischen Anfrage.

Related Concepts.

- Abschnitt 5.1.5, „XML Repräsentation von Datentypen“

Beispiel 5.4. Generic VRRequest format

5.1.5. XML Repräsentation von Datentypen

5.1.5.1. Einfache Typen

Einfache Typen werden auf folgende XML Schema Typen abgebildet:

Visual Rules Datentyp	XML Schema Typ
`String`	`xsd:string`
`Integer`	`xsd:integer`
`Float`	`xsd:decimal`
`Boolean`	`xsd:boolean`
`Date`	`xsd:date`
`Time`	`xsd:time`
`Timestamp`	`xsd:dateTime`

`String` Werte werden einfach als Text abgebildet (ohne Anführungszeichen). Zum Beispiel wird in diesem XML Fragment für ein Datenelement `name` ein String Wert `"Peter"` angegeben.

```
<name>Peter</name>
```

`Integer` Werte werden als Ganzzahlen abgebildet. Zum Beispiel wird in diesem XML Fragment 199 als der ganzzahlige Wert für ein Datenelement `seat_no` angegeben.

```
<seat_no>199</seat_no>
```

`Float` Werte werden als Gleitkommazahlen dargestellt, unter Verwendung eines Punkt als Dezimaltrennzeichen. Zum Beispiel wird in diesem XML Fragment für verschiedene Datenelemente jeweils eine Gleitkommazahl als Wert angegeben. Bitte lesen Sie die XML Schema Dokumentation für detaillierte Beschreibungen von `xsd:decimal`.

```
<price>7.45</price>
<rate>-.4453</rate>
```

`Boolean` Werte werden durch das Wort `true` oder `false` abgebildet. Der XML Typ ist `xsd:boolean`.

```
<student>true</student>
<coupon>false</coupon>
```

`Date` Werte entsprechen dem Format: `YYYY-MM-DD`. Dabei stehen die ersten vier Zeichen für das Jahr, dann zwei Zeichen für den Monat und schließlich zwei Zeichen für den Tag, wobei alle durch Bindestriche (-) getrennt sind. Bitte lesen Sie die XML Schema Dokumentation für eine detaillierte Beschreibung von `xsd:date`.

```
<show_date>2008-08-22</show_date>
```

`Time` Werte entsprechen dem Format: `hh:mm:ss`. Die ersten zwei Zeichen stehen für die Stunde, dann Minuten und Sekunden, jeweils getrennt durch Doppelpunkte (:). Kein Bestandteil kann weggelassen werden. Bitte lesen Sie die XML Schema Dokumentation für eine detaillierte Beschreibung von `xsd:time`.

```
<alarm>15:47:23</alarm>
```

`Timestamp` Werte werden entsprechend dem XML Schema Typ `xsd:dateTime` repräsentiert. Das Format ist `YYYY-MM-DDThh:mm:ss`, was das Datum und die Zeit getrennt durch den Buchstaben `T` darstellt. Kein Bestandteil darf weggelassen werden. Bitte lesen Sie die XML Schema Dokumentation für eine detaillierte Beschreibung für das Format von `xsd:dateTime`.

```
<startTimestamp>2009-05-30T09:30:10</startTimestamp>
```

Sofern Werte ausdrücklich als leer angegeben werden sollen (in Java durch `null` repräsentiert), kann dies durch Angabe das `xsi:nil` Attributes erreicht werden. Dazu ist es erforderlich, in der Anfrage auch den Namespace `xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"` zu importieren.

```
<myDate xsi:nil="true" />
```



Beachten Sie, dass die Angabe des `xsi:nil` Attributes nur für Datentypen funktioniert, die dies unterstützen. Diese Angabe ist beispielsweise wirkungslos, wenn es sich in Java um einen primitiven Datentyp wie `int` handelt, da diese ihrer Natur nach immer einen Wert haben.

5.1.5.2. Strukturen

Werte von Strukturen in einer Anfrage oder einer Antwort werden einfach durch Elemente von jedem Attribut repräsentiert. Zum Beispiel repräsentiert das folgende XML Fragment den Wert des Datenelements `customer` mit zwei Attributen `name` und `address`. Dabei hat `address` selbst ein Attribut `zip`.

```
<customer>
   <name>John Doe</name>
   <address>
      <zip>12345</zip>
   </address>
</customer>
```

5.1.5.3. Listen und Mengen

Die Werte einer Liste oder Menge in einer Anfrage oder Antwort wird durch eine Reihe von `element` Tags abgebildet.

Zum Beispiel sieht eine Liste (oder Menge) von Strings so aus:

```
<names>
   <element>John Doe</element>
   <element>Peter Pan</element>
   <element>Captain Hook</element>
</names>
```

Und dies ist eine Liste (oder Menge) von Kunden, wobei jeder Kunde die Attribute `name` und `address` hat:

```
<customers>
   <element>
      <name>John Doe</name>
      <address>
         <zip>12345</zip>
      </address>
   </element>
   <element>
      <name>Peter Pan</name>
      <address>
         <zip>99999</zip>
      </address>
   </element>
   <element>
      <name>Captain Hook</name>
      <address>
         <zip>99996</zip>
      </address>
   </element>
</customers>
```

5.1.5.4. Maps

Maps werden durch eine Reihe von `entry` Tags spezifiziert, jede davon hat zwei Elemente, `key` und `value` genannt. Zum Beispiel ist dies ein `String -> Customer` Mapping mit zwei Einträgen:

```
<customerMap>
   <entry>
      <key>8847-736-90</key>
      <value>
         <name>John Doe</name>
         <address>
            <zip>12345</zip>
         </address>
      </value>
   </entry>
   <entry>
      <key>2234-993-77</key>
      <value>
         <name>Peter Pan</name>
         <address>
            <zip>99999</zip>
         </address>
      </value>
   </entry>
</customerMap>
```

5.1.5.5. Aufzählungen

Der Wert für eine Aufzählung wird einfach durch Angabe des gewünschten Literals definiert. Im folgende Beispiel wird für eine Aufzählung für das Datenelement `bonus_card` der Wert `GOLD` angegeben:

```
<bonus_card>GOLD</bonus_card>
```

5.1.6. Abbildung des Regelmodells auf WSDL

Wird ein Regelmodell als Rule Service exportiert, so wird die WSDL Datei mit folgender Prozedur erstellt:

- Die WSDL enthält einen Service, der `<rule-model>Service` genannt wird.
- Jede exportierte Regel wird durch eine Operation (in einem portType) repräsentiert, die mit dem Regelnamen endet.

Die WSDL Datei importiert einige XML Schema Dateien. Die XML Schema Dateien werden folgendermaßen erstellt:

- Es gibt eine XML Schema Datei für jede Ebene im Regelmodell, d.h. eine für das Regelmodell selbst, eine für jedes Regelpaket und eine für jede Regel (Ablaufregel oder Entscheidungstabelle), die für das Exportieren als Rule Service gekennzeichnet wurde.
- Jede XML Schema Datei beinhaltet die Definitionen der Datentypen, die auf dieser Ebene des Regelmodells vorhanden sind.
- Die Namespace URI jedes XML Schemas ist vom Regelpaketnamen abgeleitet, d.h. Sie werden im XML Schema Namespace die gleiche Hierarchie der Regelpakete und Regeln finden.
- Die XML Schema Dateien für exportierte Regeln enthalten auch die Definitionen des Nachrichtenformats für die Input und Output Datenelemente. Anders ausgedrückt wird damit die Schnittstelle der Regel, einschließlich der Namen und Typen aller Eingabe/Ausgabe Datenelemente, beschrieben.

Weiterführende Arbeitsschritte.

- Abschnitt 4.2.5, „Anzeige der WSDL-Datei eines Rule Service“

5.2. Arbeitsschritte

5.2.1. Aufrufen eines Rule Service

Um einen Rule Service im Execution Server aufzurufen, müssen Sie eine SOAP Anfrage an den HTTP Endpunkt schicken, der in der WSDL spezifiziert wurde. Standardmässig ist dieser Endpunkt der gleiche für alle Rule Services (was es einfacher für generische Clients macht).

Das Vorgehen, um einen Web Service aufzurufen, hängt stark von der Programmiersprache oder dem Tool ab, das Sie verwenden und kann hier nicht beschrieben werden. So wird in den folgenden Beschreibungen lediglich erklärt, was technisch und im SOAP Protokoll geschieht.

1. Im Header der SOAP Nachricht liefert der Web Service Client Authentifizierungsdaten, wie im Abschnitt 5.1.1, „Authentifizierung für eine Rule Service Anfrage“ beschrieben.
2. Im Body der SOAP Nachricht spezifiziert der Web Service Client einen `VRRequest`, wie im Abschnitt 5.1.2, „Format der Rule Service Anfrage“ beschrieben. Beispielsweise kann dies so aussehen:

```
<pricing:VRRequest
        xmlns:pricing="http://www.visual-rules.com/vrpath/Movie%20Ticket%20Pricing/Pricing">
   <input>
      <auditorium_no>1</auditorium_no>
      <seat_no>199</seat_no>
      <show_date>2008-08-22</show_date>
      <coupon>true</coupon>
      <student>true</student>
      <bonus_card>GOLD</bonus_card>
   </input>
</pricing:VRRequest>
```



Nur der XML Namespace des `VRRequest` Elements wird vom Execution Server verwendet, um die Anfrage an das korrekte Regelmodell weiterzuleiten. Die URL des Service Endpunkts oder der `SOAPAction` HTTP Header sind nicht relevant für die Weiterleitung.

3. Optional kann das `target` Element als das erste Element im `VRRequest` hinzugefügt werden, um genau zu spezifizieren, welche Version des Regelmodells aufgerufen werden soll. Wenn dies weggelassen wird, ruft der Execution Server die "neueste" Version auf.

4. Wenn der Regelaufruf erfolgreich ist, erhalten Sie eine `VRResponse` wie in Abschnitt 5.1.3, „Format der Rule Service Antwort“ beschrieben. Die Ergebnisse des Regelaufrufs können im `output` Element der `VRResponse` gefunden werden. Das zusätzliche Trace-Element beinhaltet Informationen über das aufgerufene Regelmodell, die Regel und die Version.

```
<pricing:VRResponse xmlns:vr="http://www.visual-rules.com"
        xmlns:pricing="http://www.visual-rules.com/vrpath/Movie%20Ticket%20Pricing/Pricing">
   <output>
      <price>7</price>
   </output>
   <vr:trace>
      <ruleModel>Movie Ticket Pricing</ruleModel>
      <rulePath>/Movie Ticket Pricing/Pricing</rulePath>
      <version>1.0.1</version>
   </vr:trace>
</pricing:VRResponse>
```

Wenn irgendetwas fehlschlägt, erhalten Sie eine SOAP Fehlernachricht. Die Fehlernachricht enthält unter anderem einen Java Stack Trace für die Fehlerdiagnose.

Weiterführende Konzepte.

- Abschnitt 5.1.1, „Authentifizierung für eine Rule Service Anfrage“
- Abschnitt 5.1.2, „Format der Rule Service Anfrage“
- Abschnitt 5.1.3, „Format der Rule Service Antwort“

Weiterführende Arbeitsschritte.

- Abschnitt 4.2.5, „Anzeige der WSDL-Datei eines Rule Service“

Kapitel 6. Arbeiten mit Metadaten

6.1. Konzept Metadaten

Metadaten werden durch einfache Schlüssel-Wert Paare abgebildet und können zu jedem Rule Service hinzugefügt werden. In erster Linie dienen sie als erweiterte Informationen für ein Regelmodell. Beispielsweise könnte dies der Name eines bestimmten Algorithmus sein, der für eine Berechnung eingesetzt wird, oder auch einfach nur der Autor der Regeln.



Wenn Sie in einem multinationalen Team arbeiten, sollten Sie passende Namen für Metadaten wählen, damit diese von jedem verstanden werden.

Mit Hilfe von Metadaten ist es möglich eine WSDL für einen Rule Service abzuholen und der Execution Server ist in der Lage damit generische Anfragen zu einem Rule Service zu leiten. Das hängt von der eingesetzten Implementierung der `Metadata Mapper` Komponente ab. Eine Standardimplementierung einer `Metadata Mapper` Komponente ist in Abschnitt 6.2.1.1, „Standard-Metadata Mapper" beschrieben. Es ist auch möglich eine benutzerdefinierte Implementierung zu verwenden. Letzteres ist in Abschnitt 6.2.1.3, „Benutzerdefinierter Metadata Mapper" beschrieben.

6.2. Metadaten definieren

Metadaten bestehen aus einer Menge eindeutiger Schlüssel und zugehöriger Werte. Es gibt zwei Wege, um Metadaten zu bearbeiten. Sie können nach der Bereitstellung in der Webkonsole eingegeben werden (was in Abschnitt 4.2.8, „Verwaltung der Metadaten eines Rule Service" beschrieben ist) oder vor der Bereitstellung im Visual Rules Modeler. Dabei werden die Metadaten im Regelmodell gespeichert und bilden eine feste Definition, die nach der Bereitstellung nicht mehr geändert werden kann. Das garantiert, dass die Metadaten immer existieren, auch wenn die Regelbibliothek auf mehreren Servern zum Einsatz kommt. Zu beachten ist, dass es weiterhin möglich ist, zusätzliche Metadaten über die Webkonsole anzugeben.

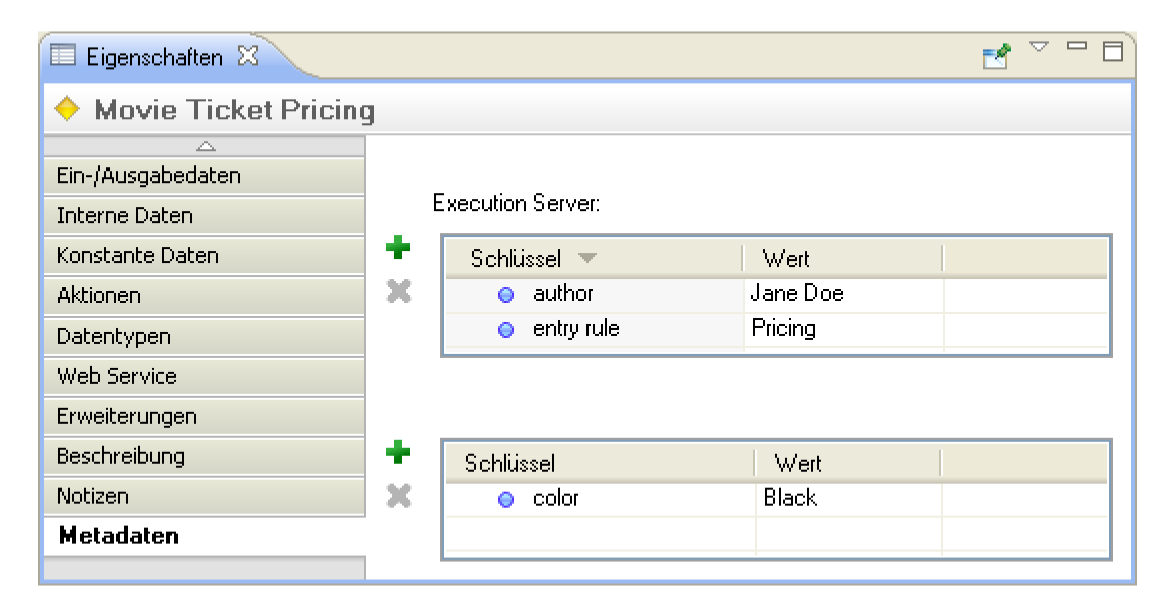
Um Metadaten im Visual Rules Modeler anzugeben führen Sie folgende Schritte aus:

1. Im **Projekt Explorer** oder **Regel Explorer**, wählen Sie das Regelmodell aus, welches als Rule Service exportiert werden soll.

2. Wechseln Sie zur Seite **Metadaten** in der Sicht **Eigenschaften**.

3. Drücken Sie den Knopf neben der **Execution Server** Tabelle. Ein Eintrag in der **Schlüssel** Spalte wird angelegt, mit einem default Namen. Der Schlüssel kann durch einen Doppelklick bearbeitet werden. Mit dem Knopf kann ein Schlüssel und sein Wert gelöscht werden.

4. Mit einem Doppelklick in die **Wert** Spalte neben dem Schlüssel kann ein Wert hinzugefügt oder bearbeitet werden.



Schlüssel und Wert müssen in einem bestimmten Format angegeben werden. Die Länge muss mindestens ein Zeichen, aber maximal 255 Zeichen sein. Leerzeichen sind innerhalb des Elements erlaubt, jedoch nicht am Anfang oder am Ende. Zu beachten ist, dass nur Schlüssel und Werte in der **Execution Server** Tabelle für die Bereitstellung verwendet werden.

6.2.1. Zuordnung von Metadaten zu Rule Services

Der Execution Server ist in der Lage, anhand von Metadaten eine WSDL zu liefern oder eine generische Anfrage zu einem Rule Service zu leiten. Dies wird dadurch bewerkstelligt, dass Metadaten einem bestimmten Rule Service zugeordnet werden. Die Komponente, die dafür verantwortlich ist, heißt `Metadata Mapper`.

Abhängig davon, welche Werte in den Metadaten definiert sind und was in der Anfrage angegeben wurde, kann es dazu kommen, dass es keine eindeutige Zuordnung gibt. Je nachdem, welches Protokoll verwendet wird, führt das in so einem Fall zu einem SOAP oder HTTP Fehler.

6.2.1.1. Standard-Metadata Mapper

Die Standardimplementierung des `Metadata Mapper` benutzt `ruleModel`, `rulePath` und optional `version` für die Weiterleitung einer Anfrage. Damit kann eine generische Anfrage anstelle einer spezifischen verwendet werden, wie das in Abschnitt 5.1.4, „Generische Rule Service Anfragen“ beschrieben wird. Abgesehen davon ist es möglich, `effectiveDate` in Kombination mit `ruleModel` und `rulePath` zur Weiterleitung einer Anfrage zu benutzen. In diesem Fall soll ein Rule Service ausgeführt werden, der zu einem bestimmten Datum gültig ist.



Der `rulePath` hat das gleiche Format, wie es in der Generischen Regel Ausführung API verwendet wird, die im Java Integration Handbuch beschrieben ist.

Der Standard-`Metadata Mapper` benutzt die Rule Service Einstellungen `gültig von` (`validFrom`) und `gültig bis` (`validTo`). Ein Rule Service ist gültig, falls sich `effectiveDate` zwischen den Grenzen von `gültig von` und `gültig bis` (inklusive) befindet. Ist eine der Grenzen nicht gesetzt, wird dies als "immer" interpretiert. Sind beide Grenzen nicht gesetzt, dann ist ein Rule Service immer gültig.

Der Hauptanwendungsfall sind Rule Services in unterschiedlichen Versionen, die zu unterschiedlichen Zeiten gültig sind. Gibt es zum Beispiel ein `Movie Ticket Pricing` in Version 1.0, das ein `gültig bis` auf 2008-12-31 gesetzt hat und es gibt ein weiteres in Version 2.0, dessen `gültig von` auf 2009-01-01 gesetzt ist, dann wird eine Anfrage mit einem `effectiveDate` auf einen Wert vor dem 2009-01-01 die Version 1.0 ausführen und im anderen Fall die Version 2.0.

Aus Gründen der Abwärtskompatibilität werden weiterhin zwei spezielle Metadaten, nämlich `effectiveFrom` und `effectiveTo` unterstützt. Jeder Wert repräsentiert ein Datum und folgt dem Format `yyyy-mm-dd`. Zu beachten ist, dass diese Metadaten nur verwendet werden, wenn weder `gültig von` noch `gültig bis` gesetzt sind.

6.2.1.2. Eine WSDL durch Angabe von Metadata abholen

Der Execution Server erlaubt es, eine WSDL für einen Rule Service durch die Angabe von Metadaten abzuholen. Die dafür benutzte URL hat die Form `http://<server>:<port>/<context-name>/services/<tenant-id>/wsdl` und Metadaten Elemente werden als Parameter angefügt. Die akzeptierten Werte hängen von der Implementierung des `Metadata Mapper` ab.

Im folgenden Beispiel wird eine Standardinstallation des Execution Servers, die unter der URL `http://localhost:8080/executionserver-6.x.y/` verfügbar ist, und ein Mandant mit ID `00ef-02394` angenommen. Wegen der Lesbarkeit wurden Zeilenumbrüche eingefügt. Normalerweise würde die URL in einer Zeile eingegeben. Die Platzhalter x und y entsprechen dabei der Versionsnummer aus dem Dateinamen der WAR Datei des Execution Servers.

```
http://localhost:8080/executionserver-6.x.y/services/00ef-02394/wsdl
?ruleModel=Movie%20Ticket%20Pricing
&rulePath=/Movie%20Ticket%20Pricing/Pricing
&effectiveDate=2008-01-01
```

In der URL im Beispiel wird eine WSDL für das Regelmodell `Movie Ticket Pricing` mit der Regel `Pricing` angefragt, das gültig ist am `2008-01-01`. Zu beachten ist, dass Sonderzeichen maskiert werden müssen, wenn sie in einer URL verwendet werden.

Beispiel 6.1. Beispiel einer URL mit dem Standard Metadata Mapper

6.2.1.3. Benutzerdefinierter Metadata Mapper

Die Anpassung des `Metadata Mapper` ist ein fortgeschrittenes Thema, da dies die Anpassung der Execution Server Installation, die Erweiterung einer abstrakten Java Klasse und die Änderung einer XML Schema Datei erfordert. Der Vorgang wird anhand eines Beispiels dargestellt.

Der angepasste `Metadata Mapper` wird mit dem Metadatum `rating` (was die Kurzform für "Rating Verfahren" ist), der `version` und dem Metadatum `rulePath` (welches die Regel spezifiziert, die das Rating Verfahren ausführt) arbeiten. Letzteres wird aus technischen Gründen benötigt, weil dies erlaubt, eine generische Anfrage zu formulieren, ohne das zu verwendende Regelmodell oder Regel angeben zu müssen. Für diesen Fall ist es also erforderlich, dass `rulePath` auf jedem Rule Service definiert wird, der auch `rating` unterstützt.

Das einzige notwendige Element in einer generischen Anfrage ist `rating`. Weil dies unter Umständen nicht ausreicht, um zu entscheiden, welcher Rule Service auszuführen ist, wird im Beispiel auch die Angabe `version` wie auch andere Metadaten verwendet.

Im ersten Schritt muss die `meta.xsd` geändert werden, um valide generische Anfragen zu erlauben. Die Schlüssel und Typen der Metadaten gehören innerhalb des `target` Elements. Der untere Ausschnitt zeigt, wie das aussehen könnte.

```
...
<xsd:complexType name="Target">
  <xsd:sequence>
   <xsd:element name="rating" type="xsd:string" />
   <!-- Jeglicher Wert erlaubt -->
   <xsd:sequence>
     <xsd:any />
   </xsd:sequence>
</xsd:sequence>
</xsd:complexType>
...
```

Als nächstes wird eine Java Klasse `RatingMetaDataMapper` erzeugt, welche die abstrakte Klasse `de.visualrules.execution.core.spi.metadata.AbstractMetaDataMapper` erweitert. Da das Object später per Reflection geladen wird, muss ein default Konstruktur aufrufbar sein. Die abstrakte Klasse fordert die Implementierung von zwei Methoden, nämlich `mapRuleModelArtifact(..)` und `mapRuleInvocationTarget(..)`. Die erste Methode wird aufgerufen, sobald eine WSDL angefragt wird und die zweite, wenn eine generische Anfrage zu einem Rule Service weitergeleitet werden soll.

Die `mapRuleModelArtifact(..)` Methode bekommt zwei Argumente übergeben: die Metadaten als `java.util.Properties` und eine Instanz des `IArtifactStorageReadAccess`. Als erstes wird

geprüft, ob ein Wert für `rating` angeben wurde, ansonsten wird eine Ausnahme geworfen. Danach wird mittels der Metadaten ein `IServiceView` gesucht. Sofern einer gefunden wurde, wird dieser zu einem `RuleModelArtifact` konvertiert und zurückgegeben.

```
public RuleModelArtifact mapRuleModelArtifact(Properties metaData,
      IArtifactStorageReadAccess artifactStorage)
      throws AmbiguousRuleModelArtifactException
{
  String rating = metaData.getProperty("rating");
  if (rating == null)
  {
    throw new IllegalArgumentException("A rating must be set.");
  }

  IServiceView serviceView = findByMetaValues(metaData, artifactStorage);
  return serviceView != null ? new RuleModelArtifact(serviceView) : null;
}
```

Die `findByMetaValues(..)` Methode benutzt die `IArtifactStorageReadAccess#listServicesWithMetaValue(..)` Methode um Rule Services zu finden, in deren Metadaten es einen passenden Wert für den Schlüssel `rating` gibt. Dies kann bei einer großen Anzahl von Services der Fall sein. Zusätzlich werden solche aussortiert, deren Metadaten nicht zu denen passen, die in der Anfrage übergeben wurden. Danach bleibt im günstigsten Fall genau ein `IServiceView` übrig, der zurückgegeben werden kann. Es gibt jedoch auch die Möglichkeit, dass es mehrere Services oder gar keine gibt. In letzterem Fall wird `null` zurückgeben, was impliziert, dass nichts gefunden wurde. Für mehrere Ergebnisse wird eine Ausnahme geworfen, die Informationen zu den gefundenen Services und ihren Metadaten enthält. Für dieses Beispiel macht dies durchaus Sinn. Die Anwendungslogik benutzt die Metadaten der Anfrage um einen passenden Rule Service zu finden. Um genauer zu werden, reicht die Angabe weiterer Werte in der Anfrage. Je spezifischer das wird, desto mehr Services können ausgefiltert werden, bis nur noch einer übrig ist.

Die Art und Weise, wie mit mehreren gefundenen Rule Services umgegangen wird, kann natürlich anders gehandhabt werden als im Beispiel. Es sollte nur sichergestellt werden, dass es das gleiche Verhalten ist wie bei der Implementierung der `mapRuleInvocationTarget(..)` Methode.

```
private static IServiceView findByMetaValues(Properties metaData,
      IArtifactStorageReadAccess artifactStorage)
      throws AmbiguousRuleModelArtifactException
{
  String ratingValue = (String) metaData.get("rating");
  if (ratingValue != null)
  {
    List services = filter(artifactStorage.listServicesWithMetaValue("rating", ratingValue),
                           metaData);
    final int size = services.size();
    if (size == 1)
    {
        return (IServiceView) services.get(0);
    }
    else if (size > 1)
    {
        IServiceView[] serviceViews = (IServiceView[]) services.toArray(
                                                  new IServiceView[size]);
        throw new AmbiguousRuleModelArtifactException(
               "There are multiple rule models that fit the requested meta values.",
                serviceViews);
    }
  }
  return null;
}
```

Die `filter(..)` Methode liefert nur die Services, deren Metadaten zur Anfrage passen, was mit der Methode `matches(..)` geprüft wird.

```
private static List filter(List services, Properties metaData)
{
   List retVal = new ArrayList(services.size());
   for (Iterator it = services.iterator(); it.hasNext();)
   {
       IServiceView serviceView = (IServiceView) it.next();
       Map metaValues = serviceView.getMetaValues();
       if (metaValues != null && matches(metaData, serviceView))
       {
           retVal.add(serviceView);
       }
   }
   return retVal;
}
```

Die `matches(..)` Methode vergleicht jedes Schlüssel-Wert Paar aus der Anfrage mit denen von jedem `IServiceView`. Die Schlüssel in den Metadaten der Anfrage müssen eine Teilmenge der auf dem Service definierten bilden. Zudem müssen die Werte übereinstimmen. Die einzige Ausnahme ist die `version`, da diese kein Metadatum ist.

```
private static boolean matches(Properties metaData, IServiceView serviceView)
{
    Map metaValues = serviceView.getMetaValues();
    for (Iterator it = metaData.entrySet().iterator(); it.hasNext();)
    {
       Map.Entry me = (Map.Entry) it.next();
       String key = (String) me.getKey();
       String value = (String) me.getValue();

       if ("version".equals(key))
       {
          if (value != null && !value.equals(serviceView.getVersion()))
          {
             return false;
          }
       }
       else if (!metaValues.containsKey(key))
       {
          return false;
       }
       else if (!matchesValue(value, (IMetaValue) metaValues.get(key)))
       {
          return false;
       }
    }
    return true;
}
```

Die `matchesValue(..)` Methode vergleicht nur die Werte unter Berücksichtigung von `null`.

```
private static boolean matchesValue(String value, IMetaValue metaValue)
{
   if (value == null)
   {
      return metaValue.getValue() == null;
   }
   else
   {
      return value.equals(metaValue.getValue());
   }
}
```

Damit ist der notwendige Code geschrieben, um eine WSDL mittels Metadaten abzuholen. Zum Beispiel ist es jetzt möglich, eine WSDL für einen Rule Service anzufragen, der ein Rating Verfahren `PD` benutzt und von `John Doe` erstellt wurde (die Platzhalter x und y entsprechen der Versionsnummer aus dem Dateinamen der WAR Datei des Execution Servers und `00ef-02394` stellt die ID des Mandanten dar).

```
http://localhost:8080/executionserver-6.x.y/services/00ef-02394/wsdl?rating=PD&author=John Doe
```

Der letzte Schritt für das Beispiel ist die Implementierung der Methode `mapRuleInvocationTarget(..)`. Diese wird aufgerufen, wenn eine generische Anfrage an den Execution Server geschickt wird. Weil hierbei etwas ausgeführt wird, spezifiziert der Rückgabewert auch die Regel, die ausgeführt werden soll. Im Beispiel wird hierfür das `rulePath` Metadatum verwendet. Die grundlegende Idee ist, dass jeder Rule Service "weiß", was seine Einstiegsregel ist. Das erlaubt die Benutzung von Metadaten für die Weiterleitung einer Anfrage, ohne dass dabei ein Regelmodell oder dessen Regel angegeben werden muss.

Der folgende Ausschnitt zeigt die Implementierung der Methode `mapRuleInvocationTarget(..)`. Auch hier wird zuerst die Existenz eines Werts für `rating` geprüft. Danach wird ein `IServiceView` für die Metadaten gesucht. Das erfordert genau die gleiche Zuordnung, wie sie in `mapRuleModelArtifact(..)` verwendet wird, weshalb die vorher geschriebene `findByMetaValues(..)` Methode wiederverwendet wird. Danach wird das Metadatum `rulePath` ausgelesen, ein `RuleInvocationTarget` erzeugt und zurückgegeben.

```
public RuleInvocationTarget mapRuleInvocationTarget(Properties metaData,
      IArtifactStorageReadAccess artifactStorage)
      throws AmbiguousRuleModelArtifactException
{
   String rating = metaData.getProperty("rating");
   if (rating == null)
   {
      throw new IllegalArgumentException("No rating provided");
   }
   IServiceView serviceView = findByMetaValues(metaData, artifactStorage);
   if (serviceView != null)
   {
      Map metaValues = serviceView.getMetaValues();
      if (metaValues != null)
      {
         IMetaValue metaValue = (IMetaValue) metaValues.get("rulePath");
         if (metaValue != null)
         {
            String rulePath = metaValue.getValue();
            RuleInvocationTarget target = new RuleInvocationTarget(serviceView, rulePath);
            return target;
         }
      }
      throw new RuntimeException("No rulePath provided in service: " + serviceView);
   }
   return null;
}
```



Dieses Beispiel beachtet nicht, ob ein Rule Service aktiv ist oder nicht. Damit lassen sich somit auch eigentlich inaktive Rule Services aufrufen. Das kann durch eine Anpassung der `filter(..)` Methode geändert werden, indem dort inaktive Services ausgefiltert werden.

Als nächstes muss der Execution Server konfiguriert werden, um das Beispiel `RatingMetaDataMapper` zu benutzen. Das wird erreicht durch die Einstellung `visualrules.executionserver.metadata.custom.mapper` wie es in Abschnitt 2.7, „Konfiguration des Execution Servers“ beschrieben ist. Der Wert ist der vollqualifizierte Klassenname von `RatingMetaDataMapper`. Mit dieser Konfiguration wird der Execution Server die Klasse beim Hochfahren laden. Es ist daher notwendig, dass sich die Klasse auf dem Klassenpfad befindet. Dies hängt vom eingesetzten Application Server ab. Beispielsweise kann dies im Tomcat erreicht werden, indem die Klasse in ein JAR gepackt und in das Verzeichnis `WEB-INF/lib` der ausgepackten Webapplikation gelegt wird.

Anhang A. Rule Service Klassenlader and -Hierarchie

A.1. Rule Service Klassenlader

Der Execution Server ist in der Lage mehrere Versionen von Rule Services gleichzeitig zu betreiben. Das wird erreicht durch die Erzeugung eines Klassenladers für jeden Rule Service. Diese Klassenlader sind voneinander isoliert um Probleme beim Klassen laden zu vermeiden.

Jeder Rule Service hat Information über erforderliche Abhängigkeiten für die Ausführung. Diese wird zum Zeitpunkt der Erstellung erzeugt und in der Webkonsole kann diese eingesehen werden. Wird ein Rule Service zum ersten Mal angefragt[1], dann wird ein Klassenlader erzeugt. Dieser lädt Klassen aus der Regelbibliothek und allen abhängigen Bibliotheken inklusive der *Visual Rules Runtime*. Falls eine Bibliothek nicht bereitgestellt wurde, wird eine Ausnahme geworfen, welche die Gruppen Id, Artefakt Id und Version der fehlenden Bibliothek enthält. Das ist ein Früherkennungs-Mechanismus der Fehler wie die *ClassNotFoundException* bei der Ausführung von Rule Services verhindert, deren Ursache schwer zu ermitteln sind.

Die Erzeugung eines Klassenladers benötigt seine Zeit. Damit dies nicht bei jeder Anfrage geschieht, werden die Klassenlader vom Execution Server in einem Cache vorgehalten. Nachfolgende Aufrufe des Rule Service werden dadurch schneller ausgeführt, da ein Klassenlader bereits verfügbar ist. Der Cache ist so lange aktiv wie der Rule Service verwendet wird. Wird innerhalb einer längeren Zeit der Rule Service nicht aufgerufen, wird der Klassenlader entfernt um Speicher frei zu geben. Wird der Rule Service danach wieder angefragt, wird der Klassenlader automatisch neu erzeugt.

A.2. Klassenlader Hierarchie

Obwohl jeder Rule Service seinen Klassenlader hat, gibt es immer auch einen vorgelagerten Klassenlader. Da auch der Anwendungsserver mehrere Klassenlader verwendet, wird dadurch eine Hierarchie gebildet.

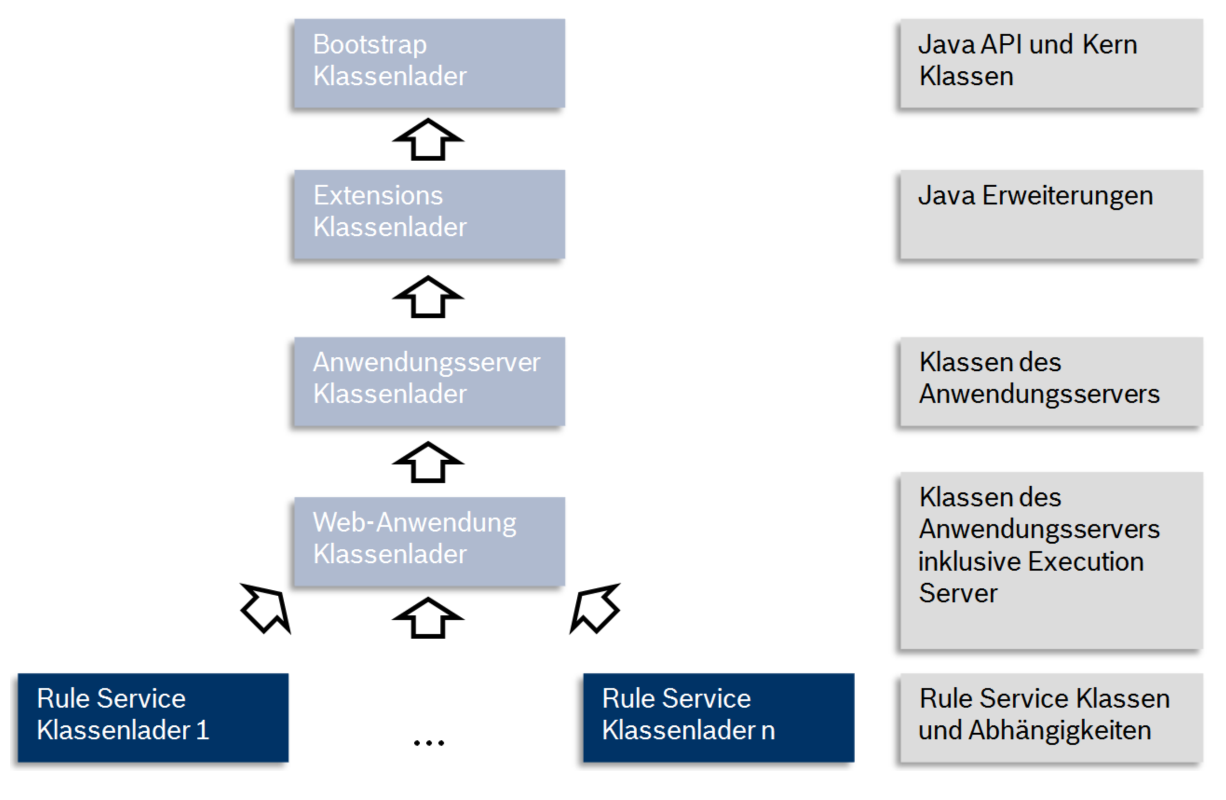


Abbildung A.1. Klassenlader Hierarchie

Die Hierarchie beeinflusst das Klassen laden. Klassen werden typischerweise vom vorgelagerten Klassenlader zuerst geladen[2]. Ist beispielsweise ein JDBC Treiber auf dem Klassenpfad des Anwendungsservers, so wird

[1] Entweder bei einem Web Service Aufruf oder der Anforderung der WSDL
[2] Manche Anwendungsserver können die Reihenfolge des Klassen ladens einstellen. Informationen dazu sollten sich in der Dokumentation des Anwendungsservers befinden.

dieser vom Anwendungsserver Klassenlader zuerst geladen. Sofern in diesem Beispiel der JDBC Treiber von einem Rule Service Klassenlader geladen werden soll, wird stattdessen die Klasse verwendet, die vom Anwendungsserver Klassenlader zuerst geladen wurde. Daher ist es empfehlenswert, den Klassenpfad und Einstellungen des Anwendungsservers genau zu prüfen.